HP-UX 参考手册

第 4 节：文件格式

HP-UX 11i v2 2004 年 9 月

第 8 卷（ 共 10 卷 ）



生产部件号： B2355-90938

E0904

© 版权所有 1983-2005 Hewlett-Packard Development Company, L.P.

法律声明

本文档中的信息如有更改，恕不另行通知。

担保

随 HP 产品及服务提供的明示性担保声明中列出了适用于此 HP 产品及服务的专用担保条款。本文中的任何内容均不构成额外的担保。HP 对本文中的技术或编辑错误以及缺漏不负任何责任。

美国政府许可

机密计算机软件。必须有 HP 授予的有效许可证，方可拥有、使用或复制本软件。根据供应商的标准商业许可证的规定，美国政府应遵守 FAR 12.211 和 12.212 中有关“商业计算机软件”、“计算机软件文档”与“商业货物技术数据”条款的规定。

附加版权声明

本文档及其所涉及的软件可能同时受到下述一项或多项版权的保护。某些单独的联机帮助页将对这些附加版权加以认可。

© 版权所有 1979, 1980, 1983, 1985-1993 Regents of the University of California

© 版权所有 1980, 1984, 1986 Novell, Inc.

© 版权所有 1985, 1986, 1988 Massachusetts Institute of Technology

© 版权所有 1986-2000 Sun Microsystems, Inc.

© 版权所有 1988 Carnegie Mellon University

© 版权所有 1989-1991 The University of Maryland

© 版权所有 1989-1993 The Open Software Foundation, Inc.

© 版权所有 1990 Motorola, Inc.

© 版权所有 1990-1992 Cornell University。

© 版权所有 1991-2003 Mentat Inc.

© 版权所有 1996 Morning Star Technologies, Inc.

© 版权所有 1996 Progressive Systems, Inc.

商标声明

Intel 和 Itanium 均为 Intel Corporation 在美国和其他国家（地区）的注册商标，使用时已受到许可。

Java 是 Sun Microsystems, Inc. 在美国的商标。

MS-DOS 和 Microsoft 是 Microsoft Corporation 在美国的注册商标。

OSF/Motif 是 The Open Group 在美国和其他国家 （地区）的商标。

UNIX 是 Open Group 的注册商标。

X Window System 是 X/Open Group 的商标。

版本说明

本文档的印刷日期和部件号指明其当前版本。印刷新版本时印刷日期会随之改变。再次印刷时可能会稍作改动，但不会更改印刷日期。新版文档将汇总从上一版本到现在所更新的所有资料。

部件号	日期；发行版；文档格式；发布形式
B2355-90931~40	2004 年 9 月； HP-UX 11i v2； PDF （10 卷）；docs.hp.com 和印刷版本。中文版。
B2355-60105	2004 年 9 月； HP-UX 11i v2； HTML （1 卷）；docs.hp.com 和 Instant Information。英文版。
B2355-90839~48	2004 年 9 月； HP-UX 11i v2； PDF （10 卷）；docs.hp.com 和印刷版本。英文版。
B2355-60103	2003 年 8 月； HP-UX 11i v2； HTML （1 卷）；docs.hp.com 和 Instant Information。英文版。
B2355-90779~87	2003 年 8 月； HP-UX 11i v2； PDF （9 卷）；docs.hp.com 和印刷版本。英文版。
B9106-90010	2002 年 6 月； HP-UX 11i v1.6； HTML （1 卷）；docs.hp.com 和 Instant Information。英文版。
B9106-90007	2001 年 6 月； HP-UX 11i v1.5； HTML （7 卷）；docs.hp.com 和 Instant Information。英文版。
B2355-90688	2000 年 12 月； HP-UX 11i v1； 9 卷。英文版。
B2355-90166	1997 年 10 月； HP-UX 11.0； 5 卷。英文版。
B2355-90128	1996 年 7 月； HP-UX 10.20； 5 卷；只有联机版本。英文版。
B2355-90052	1995 年 7 月； HP-UX 10.0； 4 卷。英文版。

印刷字体约定

本手册使用下列印刷字体约定。

audit (5)	表示 HP-UX 联机帮助页。“audit”是联机帮助页名称，“5”是该联机帮助页在 《HP-UX 参考手册》中的小节号。在网站和 Instant Information DVD 上，可能是指向该联机帮助页的热链接。在 HP-UX 命令行输入 “man audit”或 “man 5 audit”可以查看该联机帮助页。详见 *man* (1)。
系统字体	表示计算机显示的文本和系统项 (ComputerOutput)。
强调内容	第一次定义的名词和强调的内容用**黑体**表示。
键盘操作	键盘键名称。注意， Return 和 Enter 指的是同一个键。
《书名》	表示文档中引用的书籍、手册的名称，以宋体表示。
“术语”	表示文档中引用的专用术语，以宋体表示。
[ ]	格式和命令说明中的可选内容。如果内容用 “|”分隔，就必须选择其中之一。
{ }	格式和命令说明中必需的内容。如果内容用 “|”分隔，就必须选择其中之一。
...	前面的元素可以重复任意多次。
|	分隔选项列表中的项目。

前言

HP-UX 是 Hewlett-Packard Company 开发的一种操作系统，可与各种行业标准兼容。该操作系统基于 UNIX® System V Release 4 操作系统，并包括 Fourth Berkeley Software Distribution 中的重要功能。

本手册包括系统参考文档资料，即联机帮助页。单个的文档也称为联机帮助页和参考页。

简介

有关 HP-UX 以及联机帮助页的结构和格式的简介信息，请参阅 *introduction* (9) 联机帮助页。

小节简介

联机帮助页分为若干节，各节也包含 introduction 或 intro 联机帮助页，该联机帮助页介绍具体的内容。这些节有：

intro (1)	第1节：用户命令
intro (1M)	第1M节：系统管理命令
intro (2)	第2节：系统调用
intro (3C)	第3节：库函数
intro (4)	第4节：文件格式
intro (5)	第5节：其他主题
intro (7)	第7节：设备专用文件
intro (9)	第9节：简介和词汇表

第 8 卷

目录

第 4 节

第 8 卷

目录

第 4 节

目录
第 8 卷

第 4 节：文件格式

目录
第 8 卷

条目名称(节)：名称 说明

目录
第 8 卷

第 4 节

文件格式

第 4 节

文件格式

名称

intro - 文件格式简介

说明

本节概要介绍各种文件的格式。文件格式的 C **struct** 定义在可行时给定。通常，这些结构可以在目录 **/usr/include** 或 **/usr/include/sys** 中找到。

另请参阅

hier(5)、 introduction(9)。

可以通过 Web 访问 HP-UX 文档，网址为： **http://docs.hp.com** 。

名称

a.out - 汇编程序和链接编辑器输出

概要

#include <elf.h> (for ELF files)

#include <a.out.h> (for SOM files)

说明

ELF a.out

文件名 **a.out** 是链接编辑器 *ld*(1) 缺省的输出文件名。如果链接中没有任何错误，链接编辑器将创建 **a.out** 可执行文件。尽管汇编程序 *as*(1) 的输出文件的缺省文件名有所不同，但是该文件也遵循 **a.out** 文件的格式。

处理 ELF 文件的程序可以使用 *elf*(3E) 所描述的库。下面介绍文件格式的概述。有关更完整的信息，请参阅下面给出的参考资料。

```
        链接视图                      执行视图
 ________________________      ________________________
         ELF 头                         ELF 头
|______________________|      |______________________|
|       程序头表       |      |        程序头表      |
|        可选          |      |                      |
|______________________|      | _____________________|
|       节 1           |      |                      |
|______________________|      |                      |
|        . . .         |      |        段 1          |
|                      |      |                      |
|______________________|      | _____________________|
|       节 n           |      |                      |
|______________________|      |                      |
|        . . .         |      |        段 2          |
|                      |      |                      |
|______________________|      | _____________________|
|        . . .         |      |         . . .        |
|______________________|      | _____________________|
|       节头表         |      |        节头表        |
|                      |      |        可选          |
|______________________|      | _____________________|
```

ELF 头驻留在开头并保存一个描述文件组织方式的“线路图”。节用来存放链接视图的大部分对象文件信息：指令、数据、符号表、重定位信息等。段用来存放程序执行视图的对象文件信息。正如所显示的那样，一个段中可以包含一个或多个节。

程序头表（如果有）告知系统如何创建进程映像。用于创建进程映像（执行程序）的文件必须有一个程序头表；可重定位的文件则不需要。节头表中包含用来描述文件的节的信息。在表中，每个节都有一个与之相对应的条目；每个条目提供节名称、节大小等信息。在链接过程中使用的文件必须有一个节头表；其他对象文件可以有也可以没有。

尽管上图中显示程序头表紧跟在 ELF 头后面，节头表跟在节后面，但是实际文件可能会有所不同。而且，没有为节和段指定顺序。在文件中，只有 ELF 头才有固定位置。

当将 **a.out** 文件加载到内存中以执行它时，可以设置三个逻辑段：文本段、数据段（已初始化的数据后跟未初始化的数据，后者实际上全部被初始化为 0）和堆栈。文本段不能由程序写入；如果其他进程正在执行同一个 **a.out** 文件，则这些进程将共享一个文本段。

数据段从超出最后一个文本地址的下一个最大页边界开始（如果系统支持多个页大小，则“最大页”是所支持的最大大小）。在创建进程映像时，文件中用来存放文本末尾和数据开头的部分可能会出现两次。出现在数据开头的重复文本组块从不执行；它之所以重复，是为了操作系统可以在多个实际页大小中引入文件片段，而不必将数据节的开头与页边界重新对齐。因此， 第一个数据地址是超出文本末尾的下一个最大页边界与最后一个文本地址除以最大页大小的总和。如果最后一个文本地址是最大页大小的倍数，则不需要这种重复机制。堆栈可根据需要自动扩展。数据段可根据要求由 *brk*(2) 系统调用进行扩展。

SOM a.out（仅适用于 PA-RISC）

文件名 **a.out** 是汇编程序（请参阅 *as*(1) ）、编译程序和链接程序（请参阅 *ld*(1) ）的输出文件的缺省文件名。汇编程序和编译程序创建可重定位的对象文件，以便为向链接程序中输入作准备。链接程序创建可执行的对象文件以及共享库文件。

对象文件由如下内容组成：一个文件头、多个辅助文件头、空间词典、子空间词典、符号表、重定位信息、编译程序记录、空间字符串表、符号字符串表，以及初始化代码和数据的数据。并非所有的对象文件都需要所有这些节。对象文件必须以文件头开头，但是其余节不必按照任何特定的顺序排列；文件头中包含指向对象文件中其他每一节的指针。

可重定位的对象文件由汇编程序或编译程序创建，它必须至少包含以下各节：文件头、空间词典、子空间词典、符号表、重定位信息、空间字符串表、符号字符串表，以及代码和数据。它还可以包含辅助文件头和编译程序记录。可重定位的文件通常包含未解析的符号。链接程序可以合并可重定位的文件并搜索库，以生成一个可执行文件。链接程序还可用于合并可重定位的文件并生成一个新的可重定位的文件作为输出，它适合作为以后运行的链接程序的输入。

可执行文件由链接程序创建，它通常包含以下各节：文件头、 HP-UX 辅助文件头、空间词典、子空间词典、符号表、空间字符串表、符号字符串表，以及代码和数据。链接程序还可以将辅助文件头和编译程序记录从输入文件复制到输出文件。如果该文件已被去除（请参阅 *strip*(1) ），它将不包含符号表、符号字符串表或编译程序记录。可执行文件不能包含任何未解析的符号。

共享库文件由链接程序创建，它包含与可执行文件相同的节，还包含添加到共享库文件的代码节中的附加信息。该附加信息包含一个头、导出表、导入表，以及要由动态加载程序使用的动态重定位记录。

程序由两个可加载的空间组成：名为 **$TEXT$** 的不可写入的共享代码空间；名为 **$PRIVATE$** 的可写入的专用数

据空间。程序可以包含另一个名为 **$THREAD_SPECIFIC$** 的可加载的专用空间。程序可以包含其他不可加载的空间，这些空间中包含开发工具所需的数据。例如，符号调试信息包含在名为 **$DEBUG$** 或 **$PINFO$** 的空间中。链接程序将可加载的空间和不可加载的空间视为完全相同，因此，符号调试信息可以使用符号解析和重定位的所有通用性。

空间的寻址范围为 4,294,967,296 (2^32) 字节。每个可加载的空间分成四个象限，每个象限包含 1,073,741,824 (2^30) 字节。 HP-UX 操作系统将所有代码放在 **$TEXT$** 空间的第一个象限中，将所有数据放在 **$PRIVATE$** 空间的第二个象限中，将所有共享库代码放在共享内存空间的第三个象限中。

每个空间也可以分成名为子空间的逻辑单元。当链接程序合并可重定位的对象文件时，它会按名称将输入文件中的所有子空间分组，然后按照与每个子空间相关联的排序关键字排列该空间中的组。子空间在结构上并不重要；它们只是提供一种机制，以独立于多个输入文件的方式合并空间中的各个部分。下表说明了程序中的一些典型子空间：

$SHLIB_INFO$	动态加载所需的信息
$MILLICODE$	millicode 例行程序的代码
LIT	可共享的文本
$CODE$	代码
$UNWIND$	堆栈 unwind 信息
$GLOBAL$	Pascal 的外部块声明
$DATA$	静态初始化的数据
$COMMON$	FORTRAN common
BSS	未初始化的数据
$TBSS$	线程本地存储

子空间可以进行初始化，也可以不进行初始化（尽管通常只是不初始化 **BSS** 和 **$TBSS$** ）。对于初始化的子空间来说，子空间词典条目包含指向初始化数据的文件指针；而对于未初始化的子空间来说，子空间词典条目只包含用于在加载时初始化整个区域的 32 位模式。

在可重定位的文件中，初始化的代码和数据通常包含对该文件中其他位置的引用，以及对其他文件中定义的未解析符号的引用。这些引用在链接时使用重定位信息进行修补。重定位信息中的每个条目（“修正内容”）指定子空间在初始化数据中的位置，还指定与一个或两个符号有关的表达式，它定义应放在该位置的实际值。

在创建可执行文件时，链接程序会汇总 HP-UX 辅助文件头中的子空间词典。 HP-UX 程序只包含三个不同的节：一个用于代码，第二个用于初始化的数据，第三个用于未初始化的数据。按照惯例，该辅助文件头紧跟在文件头后面。

当将 **a.out** 文件加载到内存中以执行它时，可以设置三个内存区域： **a.out** 代码加载到新共享空间的第一个象限中；数据（初始化的数据后跟未初始化的数据）加载到新专用空间的第二个象限中；从接近数据空间第二象限中间部分的固定地址开始创建堆栈。

如果 **a.out** 文件使用共享库，则会将动态加载程序 **/usr/lib/dld.sl** 加载到内存中，并调用该程序，以便将该程序请求的所有共享库映射到内存中。共享库文本加载到共享内存空间的第三个象限中，共享库数据在数据空间的第二

个象限中进行分配。

此处描述的文件格式是为 HP 精准体系结构设计的所有操作系统的通用格式。因此，有一些字段和结构未在 HP-UX 上使用，或者保留供将来使用。

文件头

文件头的格式通过 **<filehdr.h>** 中的以下结构声明来描述。

```
struct header {
    short int    system_id;                    /* system id */
    short int    a_magic;                      /* magic number */
    unsigned int version_id;                   /* a.out format version */
    struct       sys_clock file_time;          /* timestamp */
    unsigned int entry_space;                  /* index of space containing entry point */
    unsigned int entry_subspace;               /* subspace index of entry */
    unsigned int entry_offset;                 /* offset of entry point */
    unsigned int aux_header_location;          /* file ptr to aux hdrs */
    unsigned int aux_header_size;              /* sizeof aux hdrs */
    unsigned int som_length;                   /* length of object module */
    unsigned int presumed_dp;                  /* DP value assumed during compilation */
    unsigned int space_location;               /* file ptr to space dict */
    unsigned int space_total;                  /* # of spaces */
    unsigned int subspace_location;            /* file ptr to subsp dict */
    unsigned int subspace_total;               /* # of subspaces */
    unsigned int loader_fixup_location;        /* space reference array */
    unsigned int loader_fixup_total;           /* # of space reference recs */
    unsigned int space_strings_location;       /* file ptr to sp. strings */
    unsigned int space_strings_size;           /* sizeof sp. strings */
    unsigned int init_array_location;          /* location of init pointers */
    unsigned int init_array_total;             /* # of init pointers */
    unsigned int compiler_location;            /* file ptr to comp recs */
    unsigned int compiler_total;               /* # of compiler recs */
    unsigned int symbol_location;              /* file ptr to sym table */
    unsigned int symbol_total;                 /* # of symbols */
    unsigned int fixup_request_location;       /* file ptr to fixups */
    unsigned int fixup_request_total;          /* # of fixups */
    unsigned int symbol_strings_location;      /* file ptr to sym strings */
    unsigned int symbol_strings_size;          /* sizeof sym strings */
    unsigned int unloadable_sp_location;       /* file ptr to debug info */
    unsigned int unloadable_sp_size;           /* size of debug info */
    unsigned int checksum;                     /* header checksum */
```

```
};
```

时间戳是如下所示的双字结构。如果未使用，则这两个字段均为零。

```
struct sys_clock {
   unsigned int secs;
   unsigned int nanosecs;
};
```

辅助文件头

辅助文件头包含在文件的一个连续区域中，可由文件头中的指针来定位。辅助文件头可用于两个目的：第一个目的是将用户的版本和版权字符串附加到对象文件中；第二个目的是包含加载可执行程序所需的信息。在可执行程序中， HP-UX 辅助文件头必须位于所有其他辅助文件头前面。以下声明可在 **<aouthdr.h>** 中找到。

```
struct aux_id {
   unsigned int mandatory : 1;           /* linker must understand aux hdr info */
   unsigned int copy : 1;                /* copy aux hdr without modification */
   unsigned int append : 1;              /* merge multiple entries of same type */
   unsigned int ignore : 1;              /* ignore aux hdr if type unknown */
   unsigned int reserved : 12;           /* reserved */
   unsigned int type : 16;               /* aux hdr type */
   unsigned int length;                  /* sizeof rest of aux hdr */
};

/* Values for the aux_id.type field */
#define HPUX_AUX_ID                      4
#define VERSION_AUX_ID                   6
#define COPYRIGHT_AUX_ID                 9
#define SHLIB_VERSION_AUX_ID             10

struct som_exec_auxhdr {                 /* HP-UX auxiliary header */
   struct  aux_id som_auxhdr;            /* aux header id */
   long    exec_tsize;                   /* text size */
   long    exec_tmem;                    /* start address of text */
   long    exec_tfile;                   /* file ptr to text */
   long    exec_dsize;                   /* data size */
   long    exec_dmem;                    /* start address of data */
   long    exec_dfile;                   /* file ptr to data */
   long    exec_bsize;                   /* bss size */
   long    exec_entry;                   /* address of entry point */
   long    exec_flags;                   /* loader flags */
   long    exec_bfill;                   /* bss initialization value */
};
```

```
/* Values for exec_flags */
#define TRAP_NIL_PTRS    01

struct user_string_aux_hdr {                /* Version string auxiliary header */
    struct aux_id header_id;                /* aux header id */
    unsigned int  string_length;            /* strlen(user_string) */
    char          user_string[1];           /* user-defined string */
};

struct copyright_aux_hdr {                  /* Copyright string auxiliary header */
    struct aux_id  header_id;               /* aux header id */
    unsigned int   string_length;           /* strlen(user_string) */
    char           copyright[1];            /* user-defined string */
};

struct shlib_version_aux_hdr {
    struct aux_id  header_id;               /* aux header id */
    short          version;                 /* version number */
};
```

空间词典

空间词典由一系列空间记录组成，如 **<spacehdr.h>** 中所定义。

```
struct space_dictionary_record {
    union name_pt name;                     /* index to space name */
    unsigned int is_loadable: 1;            /* space is loadable */
    unsigned int is_defined: 1;             /* space is defined within file */
    unsigned int is_private: 1;             /* space is not sharable */
    unsigned int has_intermediate_code: 1;  /* contains intermediate code */
    unsigned int is_tspecific: 1;           /* space is $thread_specific$ */
    unsigned int reserved: 11;              /* reserved */
    unsigned int sort_key: 8;               /* sort key for space */
    unsigned int reserved2: 8;              /* reserved */
    int          space_number;              /* space index */
    int          subspace_index;            /* index to first subspace */
    unsigned int subspace_quantity;         /* # of subspaces in space */
    int          loader_fix_index;          /* index into loader fixup array */
    unsigned int loader_fix_quantity;       /* # of loader fixups in space */
    int          init_pointer_index;        /* index into init pointer array */
    unsigned int init_pointer_quantity;     /* # of init ptrs */
};
```

空间名称的字符串包含在空间字符串表中，后者可由文件头中的指针来定位。在空间字符串表中，每个条目的前

面有一个 4 字节整数（定义字符串的长度），而且每个条目的末尾有一到五个空字符（填充字符串，直到字边界）。该表的索引与表的开头有关，它指向字符串的第一个字节（而不是前导的长度字）。下面定义的并集用于所有此类的字符串指针；字符指针是为将字符串表读入内存中的程序定义的，并且旨在重定位空间记录的内存中副本。

```
union name_pt {
  char         *n_name;
  unsigned int n_strx;
};
```

子空间词典

子空间词典由一系列子空间记录组成，如 **<scnhdr.h>** 中所定义。子空间名称的字符串包含在空间字符串表中。

```
struct subspace_dictionary_record {
  int          space_index;                /* index into space dictionary */
  unsigned int access_control_bits: 7;     /* access and priv levels of subsp */
  unsigned int memory_resident: 1;         /* lock in memory during exec */
  unsigned int dup_common: 1;              /* duplicate data symbols allowed */
  unsigned int is_common: 1;               /* initialized common block */
  unsigned int is_loadable: 1;             /* subspace is loadable */
  unsigned int quadrant: 2;                /* quadrant in space subsp should reside in */
  unsigned int initially_frozen: 1;        /* lock in memory when OS booted */
  unsigned int is_first: 1;                /* must be first subspace */
  unsigned int code_only: 1;               /* subspace contains only code */
  unsigned int sort_key: 8;                /* subspace sort key */
  unsigned int replicate_init: 1;          /* init values to be replicated to fill subsp len */
  unsigned int continuation: 1;            /* subspace is a continuation */
  unsigned int is_tspecific: 1;            /* subspace contains TLS */
  unsigned int reserved: 5;                /* reserved */
  int          file_loc_init_value;        /* file location or init value */
  unsigned int initialization_length;      /* length of initialization */
  unsigned int subspace_start;             /* starting offset */
  unsigned int subspace_length;            /* total subspace length */
  unsigned int reserved2: 16;              /* reserved */
  unsigned int alignment: 16;              /* alignment required */
  union name_pt name;                      /* index of subspace name */
  int          fixup_request_index;        /* index to first fixup */
  unsigned int fixup_request_quantity;     /* # of fixup requests */
};
```

符号表

符号表由一系列条目组成，这些条目通过 **<syms.h>** 中的如下所示的结构来描述。符号和限定符名称的字符串包含在符号字符串表中，该表的结构与空间字符串表的结构相同。

```
struct symbol_dictionary_record {
    unsigned int   hidden: 1;              /* symbol not visible to loader */
    unsigned int   secondary_def: 1;       /* secondary def symbol */
    unsigned int   symbol_type: 6;         /* symbol type */
    unsigned int   symbol_scope: 4;        /* symbol value */
    unsigned int   check_level: 3;         /* type checking level */
    unsigned int   must_qualify: 1;        /* qualifier required */
    unsigned int   initially_frozen: 1;    /* lock in memory when OS booted */
    unsigned int   memory_resident: 1;     /* lock in memory during exec */
    unsigned int   is_common: 1;           /* common block */
    unsigned int   dup_common: 1;          /* duplicate data symbols allowed */
    unsigned int   xleast: 2;              /* MPE-only */
    unsigned int   arg_reloc: 10;          /* parameter relocation bits */
    union name_pt  name;                   /* index to symbol name */
    union name_pt  qualifier_name;         /* index to qual name */
    unsigned int   symbol_info;            /* subspace index */
    unsigned int   symbol_value;           /* symbol value */
};

/* Values for symbol_type */
#define ST_NULL       0                    /* unused symbol entry */
#define ST_ABSOLUTE   1                    /* non-relocatable symbol */
#define ST_DATA       2                    /* initialized data symbol */
#define ST_CODE       3                    /* generic code symbol */
#define ST_PRI_PROG   4                    /* program entry point */
#define ST_SEC_PROG   5                    /* secondary prog entry point*/
#define ST_ENTRY      6                    /* procedure entry point */
#define ST_STORAGE    7                    /* storage request */
#define ST_STUB       8                    /* MPE-only */
#define ST_MODULE     9                    /* Pascal module name */
#define ST_SYM_EXT    10                   /* symbol extension record */
#define ST_ARG_EXT    11                   /* argument extension record */
#define ST_MILLICODE 12                    /* millicode entry point */
#define ST_PLABEL     13                   /* MPE-only */
#define ST_OCT_DIS    14                   /* Used by OCT only--ptr to translated code */
#define ST_MILLI_EXT 15                    /* address of external millicode */
```

```
#define ST_TSTORAGE  16              /* TLS common symbol */

/* Values for symbol_scope */
#define SS_UNSAT     0               /* unsatisfied reference */
#define SS_EXTERNAL  1               /* import request to external symbol */
#define SS_LOCAL     2               /* local symbol */
#define SS_UNIVERSAL 3               /* global symbol */
```

符号值的含义取决于符号的类型。对于代码符号（常规代码、程序入口点、过程入口点和 millicode 入口点），符号值的两个低位对执行权限级别进行编码，执行特权级别不在 HP-UX 中使用，但它通常设置为 3。已屏蔽掉这些位的符号值就是符号的地址（总是为 4 的倍数）。对于数据符号，符号值只是符号的地址。对于线程本地存储符号（非通用符号），符号值是库中或可执行文件中线程本地存储的偏移量，如果符号位于可重定位的对象文件中，则符号值是符号的大小。对于存储请求和线程本地存储通用符号，符号值是所请求的字节数；链接程序为 **BSS** 或 **$TBSS$** 子空间中每个符号的最大请求分配空间，但为该符号找到了本地或通用符号（这种情况下，存储请求被视为不满意的引用来处理）时例外。

如果可重定位的文件是用参数类型检查功能进行编译的，则扩展记录位于用来定义和引用过程入口点和全局变量的符号的后面。第一个扩展记录 “符号扩展记录”定义返回值或全局变量的类型，对于过程或函数来说，还定义参数的数量和前三个参数的类型。如果需要多个参数类型描述符，则它们后面跟有一个或多个 “参数扩展记录”，每个记录包含四个以上的描述符。如果检查级别为 0，则指定不进行类型检查；其后面没有扩展记录。如果检查级别为 1 或更大，则指定对返回值或全局变量类型进行检查。如果检查级别为 2 或更大，则指定对参数的数量进行检查；如果检查级别为 3，则指定对每个参数的类型进行检查。链接程序在解析符号引用时，在不满意的符号和本地或通用符号之间执行所请求级别的类型检查。

```
union arg_descriptor {
  struct {
  unsigned int   reserved: 3;        /* reserved */
  unsigned int   packing: 1;         /* packing algorithm used */
  unsigned int   alignment: 4;       /* byte alignment */
  unsigned int   mode: 4;            /* type of descriptor and its use */
  unsigned int   structure: 4;       /* structure of symbol */
  unsigned int   hash: 1;            /* set if arg_type is hashed */
  int            arg_type: 15;       /* data type */
  } arg_desc;
  unsigned int   word;
};

struct symbol_extension_record {
  unsigned int   type: 8;            /* always ST_SYM_EXT */
  unsigned int   max_num_args: 8;    /* max # of parameters */
  unsigned int   min_num_args: 8;    /* min # of parameters */
  unsigned int   num_args: 8;        /* actual # of parameters */
```

```
   union arg_descriptor symbol_desc;           /* symbol type desc. */
   union arg_descriptor argument_desc[3];      /* first 3 parameters */
};

struct argument_desc_array {
   unsigned int    type: 8;                    /* always ST_ARG_EXT */
   unsigned int    reserved: 24;               /* reserved */
   union arg_descriptor argument_desc[4];      /* next 4 parameters */
};
```

arg_descriptor 中的 **alignment** 字段指示数据的最小对齐值，值 **n** 表示 2^**n** 字节对齐。下表给出了 **arg_descriptor** 中 **mode** 、 **structure** 和 **arg_type** （如果未对数据类型进行散列处理）字段的值。

值	模式	结构	参数类型
0	任意	任意	任意
1	值参数	标量	void
2	引用参数	数组	带符号的字节
3	值结果	结构	不带符号的字节
4	名称	指针	带符号的短型
5	变量	长 ptr	不带符号的短型
6	函数返回	C 字符串	带符号的长型
7	过程	Pascal 字符串	不带符号的长型
8	长引用参数	过程	带符号的双字
9		函数	不带符号的双字
10		标签	短实数
11			实数
12			长实数
13			短复数
14			复数
15			长复数
16			集合小数
17			结构/数组

对于过程入口点，参数重定位位定义格式参数和返回值的位置。通常，参数列表的前四个字传入常规寄存器 (**r26-r23**) 而不是堆栈，返回值在 **r29** 中返回。该范围内的浮点参数传入浮点寄存器 (**fr4-fr7**) ，浮点值在 **fr4** 中返回。参数重定位位由五对位组成，这些位描述参数列表的前四个字和返回值。最左边的那一对位描述第一个参数字，最右边的那一对位描述返回值。下表说明了这些位的含义。

位	含义
00	没有参数或返回值
01	常规寄存器中的参数或返回值
10	浮点寄存器中的参数或返回值
11	双精度浮点值

对于双精度浮点参数，奇数参数字应被标记为 **11** ，偶数参数字应被标记为 **10** 。双精度返回值只被标记为 **11** 。

每个过程调用会用一组类似的位进行标记（请参阅下面的“重定位信息”），以便链接程序可以将每个调用与所期望的过程入口点相匹配。如果过程调用与入口点不匹配，则链接程序会创建一个存根，该存根可根据需要重定位参数和返回值。

重定位信息

每个初始化的子空间定义一个修正范围，这些修正内容应用于该子空间中的数据。修正请求与需要重定位或包含不满意符号引用的每个字相关联。对于在 800 系列系统上由 HP-UX 3.0 版之前的版本创建的可重定位的对象文件，每个修正请求都是五字结构，该结构描述要在链接时修补的代码或数据字。由 3.0 版或更高版本创建的对象文件包含长度可变的修正请求，这些请求描述子空间的每个字节。文件头中的 *version_id* 字段能够区分这两种格式；常量 **VERSION_ID** 可在较早的对象文件中找到，常量 **NEW_VERSION_ID** 可在较新的对象文件中找到。

在较早的对象文件中，修正功能可以计算包含零个、一个或两个符号以及一个常量的表达式，然后从计算结果中提取一个位字段，并将这些位以多种不同格式中的任一格式进行存放，这些格式与精准体系结构指令集相对应。子空间词典条目中的 *fixup_request_index* 字段将索引编制到由文件头定义的修正请求区域， *fixup_request_quantity* 字段是指用于该子空间的修正请求的数量。修正请求的结构包含在 **<reloc.h>** 中。

```
struct fixup_request_record {
   unsigned int  need_data_ref: 1;     /* reserved */
   unsigned int  arg_reloc: 10;        /* parameter relocation bits */
   unsigned int  expression_type: 5;   /* how to compute value */
   unsigned int  exec_level: 2;        /* reserved */
   unsigned int  fixup_format: 6;      /* how to deposit bits */
   unsigned int  fixup_field: 8;       /* field to extract */
   unsigned int  subspace_offset;      /* subspace offset of word */
   unsigned int  symbol_index_one;     /* index of first symbol */
   unsigned int  symbol_index_two;     /* index of second symbol */
   int           fixup_constant;       /* constant */
};

/* Values for expression_type */
#define e_one     0          /* symbol1 + constant */
#define e_two     1          /* symbol1 - symbol2 + constant */
#define e_pcrel   2          /* symbol1 - pc + constant */
#define e_con     3          /* constant */
#define e_plabel  7          /* symbol1 + constant */
```

```
#define e_abs     18           /* absolute, 1st sym index is address */

/* Values for fixup_field (assembler mnemonics shown) */
#define e_fsel    0            /* F': no change */
#define e_lssel   1            /* LS': inverse of RS' */
#define e_rssel   2            /* RS': rightmost 11 bits, signed */
#define e_lsel    3            /* L': leftmost 21 bits */
#define e_rsel    4            /* R': rightmost 11 bits */
#define e_ldsel   5            /* LD': inverse of RD' */
#define e_rdsel   6            /* RD': rightmost 11 bits, filled left with ones */
#define e_lrsel   7            /* LR': L' with "rounded" constant */
#define e_rrsel   8            /* RR': R' with "rounded" constant */
#define e_nsel    9            /* N1': set all bits to zero: for id of 3-inst code gen sequence */

/* Values for fixup_format (typical instructions shown) */
#define i_exp14   0            /* 14-bit immediate (LDW, STW) */
#define i_exp21   1            /* 21-bit immediate (LDIL, ADDIL) */
#define i_exp11   2            /* 11-bit immediate (ADDI, SUBI) */
#define i_rel17   3            /* 17-bit pc-relative (BL) */
#define i_rel12   4            /* 12 bit pc-relative (COMBT, COMBF, etc.) */
#define i_data    5            /* whole word */
#define i_none    6
#define i_abs17   7            /* 17-bit absolute (BE, BLE) */
#define i_milli   8            /* 17-bit millicode call (BLE) */
#define i_break   9            /* reserved (no effect on HP-UX) */
```

在较新的对象文件中，重定位条目由字节流组成。子空间词典条目中的 *fixup_request_index* 字段是由文件头定义的修正词典的字节偏移量， *fixup_request_quantity* 字段为该子空间定义修正请求流的长度（以字节为单位）。每个修正请求的第一个字节（操作码）都标识该请求并确定该请求的长度。

通常，修正流是一系列链接程序指令，用于管理链接程序如何将数据放在 **a.out** 文件中。某些修正请求会导致链接程序将一个或多个字节从输入子空间毫无变化地复制到输出子空间，而另一些修正请求会指示链接程序重定位字或解析外部引用。还有一些请求指示链接程序在输出子空间中插入零，或者将区域保持未初始化状态，而不从输入子空间复制任何数据；其他请求描述代码中的点，而不向输出文件中添加任何新数据。

包含文件 **<reloc.h>** 为每个主操作码定义常量。许多修正请求使用一定范围的操作码；仅定义范围开头的一个常量。下面描述了每个修正请求的含义，并在随后的表中描述了每个修正的操作码范围和参数。

R_NO_RELOCATION 复制 L 字节，不进行重定位。

R_ZEROES 向输出子空间中插入 L 零字节。

R_UNINIT 跳过输出子空间中的 L 字节。

R_RELOCATION 复制一个数据字，并进行重定位。假定该数据字包含与其自己的子空间相关的 32 位指针。

R_DATA_ONE_SYMBOL 复制一个数据字，并且相对于符号索引为 S 的外部符号进行重定位。

R_DATA_PLABEL 将一个数据字作为 32 位过程标签进行复制，来引用符号 S。该数据字的原始内容应为 0（无静态链接）或 2（需要静态链接）。

R_SPACE_REF 将一个数据字作为空格引用进行复制。当前不支持该修正请求。

R_REPEATED_INIT 从输入子空间复制 L 字节，并复制该数据以填充输出子空间中的 M 字节。

R_PCREL_CALL 复制一个指令字，并进行重定位。假定该指令字是一个与 pc 相关的过程调用指令（例如， **BL** ）。目标过程由符号 S 来标识，参数重定位位是 R。

R_ABS_CALL 复制一个指令字，并进行重定位。假定该指令字是一个绝对过程调用指令（例如， **BLE** ）。目标过程由符号 S 来标识，参数重定位位是 R。

R_DP_RELATIVE 复制一个指令字，并进行重定位。假定该指令字是与 dp 相关的加载或存储指令（例如， **ADDIL** 、 **LDW** 和 **STW** ）。目标符号由符号 S 来标识。链接程序导致了符号 S 的值与符号 **$global$** 的值之间的差异。按照惯例， **$global$** 的值总是包含在寄存器 27 中。在指令的位移字段中，指令可以有一个小常量。

R_DLT_REL 复制一个指令字，并进行重定位。假定该指令字是与寄存器 18 相关的加载或存储指令（例如， **LDW** 、 **LDO** 和 **STW** ）。目标符号由符号 S 来标识。链接程序为符号 S 计算与寄存器 18 相关的链接表偏移量（在位置无关代码中保留为链接表指针）。

R_CODE_ONE_SYMBOL 复制一个指令字，并进行重定位。假定该指令字是引用符号 S 的指令（例如， **LDIL** 、 **LDW** 和 **BE** ）。在指令的位移字段中，指令可以有一个小常量。

R_MILLI_REL 复制一个指令字，并进行重定位。假定该指令字是一个短 millicode 调用指令（例如， **BLE** ）。链接程序导致了目标符号 S 的值与模块符号表中符号 1 的值之间的差异。按照惯例，符号 1 的值应当以前已加载到 BLE 指令所使用的基寄存器中。在指令的位移字段中，指令可以有一个小常量。

R_CODE_PLABEL 复制一个指令字，并进行重定位。假定该指令字是形成过程标签的代码序列（例如， **LDIL** 和 **LDO** ）的一部分，并引用符号 S。 **LDO** 指令应当在其位移字段中包含值 0（无静态链接）或 2（需要静态链接）。

R_BREAKPOINT 有条件地复制一个指令字。在 HP-UX 上，链接程序总是将该指令字替换为 **NOP** 指令。

R_ENTRY 定义过程入口点。堆栈的 unwind 位 (U) 和帧大小 (F) 记录在堆栈的 unwind 描述符中。

R_ALT_ENTRY 定义备用的过程入口点。

R_EXIT 定义过程出口点。

R_BEGIN_TRY 定义尝试（或恢复）区域的开头。

R_END_TRY 定义尝试（或恢复）区域的末尾。偏移量 R 定义从该区域的末尾到恢复块的开头之间的距离（以字节为单位）。

R_BEGIN_BRTAB 定义分支表的开头。

R_END_BRTAB 定义分支表的末尾。

R_AUX_UNWIND 定义辅助 unwind 表。 **CN** 是对编译单元字符串表的开头进行标记的符号的符号索引。 **SN** 是范围名称字符串的相对于 **CN** 符号的偏移量。 **SK** 是一个指定范围种类的整数。

R_STATEMENT 定义语句编号 N 的开头。

R_SEC_STATEMENT 定义辅助语句编号 N 的开头。

R_DATA_EXPR 从表达式堆栈中取出一个字，并将一个数据字从输入子空间复制到输出子空间，从而将取出的值添加到输出子空间中。

R_CODE_EXPR 从表达式堆栈中取出一个字，并将一个指令字从输入子空间复制到输出子空间，从而将取出的值添加到该指令的位移字段中。

R_FSEL 为下一个修正请求使用 F′ 字段选择器，而不使用适于该指令的缺省选择器。

R_LSEL 为下一个修正请求使用 L 类字段选择器，而不使用适于该指令的缺省选择器。根据当前的舍入模式，可以使用 L′、LS′、LD′ 或 LR′。

R_RSEL 为下一个修正请求使用 R 类字段选择器，而不使用适于该指令的缺省选择器。根据当前的舍入模式，可以使用 R′、RS′、RD′ 或 RR′。

R_N_MODE 选择向下舍入模式 (L′/R′)。在每个子空间的开头，这是缺省模式。只有在明确更改此设置或者到达子空间的末尾时，此设置才失效。

R_S_MODE 选择“舍入到最近的页”模式 (LS′/RS′)。只有在明确更改此设置或者到达子空间的末尾时，此设置才失效。

R_D_MODE 选择向上舍入模式 (LD′/RD′)。只有在明确更改此设置或者到达子空间的末尾时，此设置才失效。

R_R_MODE 选择“用调整过的常量向下舍入”模式 (LR′/RR′)。只有在明确更改此设置或者到达子空间的末尾时，此设置才失效。

R_DATA_OVERRIDE 为下一个修正请求用常量 V 代替输入子空间内数据字或指令中的常量。

R_TRANSLATED 切换“已转换”模式。该修正请求仅由链接程序在可重定位的链接过程中生成，以指明最初从旧格式的可重定位对象文件中读取的子空间。

R_COMP1

堆栈操作。该修正请求的第二个字节包含一个辅助操作码。在下面的说明中，A 是指堆栈的顶部， B 是指堆栈上的下一个项目。堆栈上的所有项目都被视为带符号的 32 位整数。

R_PUSH_PCON1	压入（正）常量 V。
R_PUSH_DOT	压入当前的虚拟地址。
R_MAX	取出 A 和 B，然后压入 A 和 B 中的较大值。
R_MIN	取出 A 和 B，然后压入 A 和 B 中的较小值。
R_ADD	取出 A 和 B，然后压入 A 与 B 的和。
R_SUB	取出 A 和 B，然后压入 B 减 A 所得到的值。
R_MULT	取出 A 和 B，然后压入 A 与 B 的乘积。
R_DIV	取出 A 和 B，然后压入 B 除以 A 所得到的值。
R_MOD	取出 A 和 B，然后压入 B 除以 A 所得到的余数。
R_AND	取出 A 和 B，然后压入 A 与 B 进行“与”运算所得到的值。
R_OR	取出 A 和 B，然后压入 A 与 B 进行“或”运算所得到的值。
R_XOR	取出 A 和 B，然后压入 A 与 B 进行“异或”运算所得到的值。
R_NOT	将 A 替换为它的补数。
R_LSHIFT	如果 C = 0，则取出 A 和 B，然后压入 B << A。否则，将 A 替换为 A << C。
R_ARITH_RSHIFT	如果 C = 0，则取出 A 和 B，然后压入 B >> A。否则，将 A 替换为 A >> C。移位是通过符号扩展来进行的。
R_LOGIC_RSHIFT	如果 C = 0，则取出 A 和 B，然后压入 B >> A。否则，将 A 替换为 A >> C。移位是通过填充零来进行的。
R_PUSH_NCON1	压入（负）常量 V。

R_COMP2

更多的堆栈操作。

R_PUSH_PCON2	压入（正）常量 V。
R_PUSH_SYM	压入符号 S 的值。
R_PUSH_PLABEL	压入符号 S 的过程标签的值。静态链接位是 L。
R_PUSH_NCON2	压入（负）常量 V。

R_COMP3

更多的堆栈操作。

R_PUSH_PROC	压入过程入口点 S 的值。参数重定位位是 R。
R_PUSH_CONST	压入常量 V。

R_PREV_FIXUP

链接程序保持最后四个唯一多字节修正请求的队列。这是与队列上的一个请求相同的修正请求的缩写。队列索引 X 引用这四个请求中的一个请求；如果 X = 0，则引用最近的请求。该修正请求的副作用是，所引用的修正将移到队列的前面。

R_N0SEL 指明将随后的修正应用于用来访问数据的三指令序列中的第一个指令，三指令序列由编译程序生成，以便启用对共享库数据的导入。

R_N1SEL 为下一个修正请求使用 (N′) 字段选择器。这指明要将零位用于指令的位移。该修正用于标识访问数据时所使用的三指令序列（用于导入共享库数据）。

R_LINETAB 定义行表的开头。 CU 是对行表的开头进行标记的符号的符号索引。 SM 是相对于 CU 符号的偏移量。 ES 为当前的行表指定版本信息。

R_LINETAB_ESC 定义要输入行表中的转义条目。 ES 指定在该表中输入的转义条目。 M 指定要解释为原始 8 位表数据的 R_STATEMENT 修正的数量。

R_LTP_OVERRIDE 覆盖随后的修正，在构建共享库时，它应当为 R_DATA_ONE_SYMBOL 修正，以便复制一个数据字，而不进行重定位。将复制与链接表指针有关的符号的绝对字节偏移量。如果链接程序正在构建完整的可执行文件，则将复制绝对虚拟地址。

R_COMMENT 用于将注释信息从编译程序传递到链接程序的修正。该修正有一个可被应用程序跳过和忽略的 5 字节参数。

R_TP_OVERRIDE 覆盖以下任一修正后面显示的一个修正： R_DP_RELATIVE、R_DLT_REL 或 R_DATA_ONE_SYMBOL，以便在修正指令时使用线程本地存储偏移量。该修正还用于捕获线程本地存储符号的不匹配信息。

R_RESERVED 将保留该范围内的修正，供编译程序和链接程序在内部使用。

下表说明了每一范围操作码的助记符修正请求的类型和长度以及参数信息。在参数列中，符号 D 是指操作码与该表条目所描述的范围的开头之间的区别；符号 B1、B2、B3 和 B4 分别是指该修正请求之后的第一个、第二个、第三个或第四个字节的值。

助记符	操作码	长度	参数
R_NO_RELOCATION	0-23	1	***L = (D+1) * 4***
	24-27	2	***L = (D<<8 + B1 + 1) * 4***
	28-30	3	***L = (D<<16 + B2 + 1) * 4***
	31	4	***L = B3 + 1***
R_ZEROES	32	2	***L = (B1 + 1) * 4***
	33	4	***L = B3 + 1***
R_UNINIT	34	2	***L = (B1 + 1) * 4***
	35	4	***L = B3 + 1***
R_RELOCATION	36	1	无
R_DATA_ONE_SYMBOL	37	2	***S = B1***
	38	4	***S = B3***
R_DATA_PLABEL	39	2	***S = B1***
	40	4	***S = B3***

R_SPACE_REF	41	1	无
R_REPEATED_INIT	42	2	***L = 4*** ; ***M = (B1 + 1) * 4***
	43	3	***L = (B1 + 1) * 4*** ; ***M = (B1 + 1) * L***
	44	5	***L = (B1 + 1) * 4*** ; ***M = (B3 + 1) * 4***
	45	8	***L = B3 + 1*** ; ***M = B4 + 1***
R_PCREL_CALL	48-57	2	***R = rbits1(D)*** ; ***S = B1***
	58-59	3	***R = rbits2(D<<8 + B1)*** ; ***S = B1***
	60-61	5	***R = rbits2(D<<8 + B1)*** ; ***S = B3***
R_ABS_CALL	64-73	2	***R = rbits1(D)*** ; ***S = B1***
	74-75	3	***R = rbits2(D<<8 + B1)*** ; ***S = B1***
	76-77	5	***R = rbits2(D<<8 + B1)*** ; ***S = B3***
R_DP_RELATIVE	80-111	1	***S = D***
	112	2	***S = B1***
	113	4	***S = B3***
R_DLT_REL	120	2	***S = B1***
	121	4	***S = B3***
R_CODE_ONE_SYMBOL	128-159	1	***S = D***
	160	2	***S = B1***
	161	4	***S = B3***
R_MILLI_REL	174	2	***S = B1***
	175	4	***S = B3***
R_CODE_PLABEL	176	2	***S = B1***
	177	4	***S = B3***
R_BREAKPOINT	178	1	无
R_ENTRY	179	9	***U,F = B8*** （U 是 37 位；F 是 27 位）
	180	6	***U = B5 >> 3; F = pop A***
R_ALT_ENTRY	181	1	无
R_EXIT	182	1	无
R_BEGIN_TRY	183	1	无
R_END_TRY	184	1	***R = 0***
	185	2	***R = sign_extend(B1) * 4***
	186	4	***R = sign_extend(B3) * 4***
R_BEGIN_BRTAB	187	1	无
R_END_BRTAB	188	1	无
R_STATEMENT	189	2	***N = B1***
	190	3	***N = B2***
	191	4	***N = B3***
R_DATA_EXPR	192	1	无

R_CODE_EXPR	193	1	无
R_FSEL	194	1	无
R_LSEL	195	1	无
R_RSEL	196	1	无
R_N_MODE	197	1	无
R_S_MODE	198	1	无
R_D_MODE	199	1	无
R_R_MODE	200	1	无
R_DATA_OVERRIDE	201	1	***V = 0***
	202	2	***V = sign_extend(B1)***
	203	3	***V = sign_extend(B2)***
	204	4	***V = sign_extend(B3)***
	205	5	***V = B4***
R_TRANSLATED	206	1	无
R_AUX_UNWIND	207	12	***CU,SN,SK = B11*** （CU 是 24 位；SN 是 32 位；SK 是 32 位）
R_COMP1	208	2	***OP = B1***； ***V = OP & 0x3f***； ***C = OP & 0x1f***
R_COMP2	209	5	***OP = B1***； ***S = B3***； ***L = OP & 1***； ***V = ((OP & 0x7f) << 24) | S***
R_COMP3	210	6	***OP = B1***； ***V = B4***； ***R = ((OP & 1) << 8) | (V >> 16)***； ***S = V & 0xffffff***
R_PREV_FIXUP	211-214	1	***X = D***
R_N0SEL	216	1	无
R_N1SEL	217	1	无
R_SEC_STMT	215	1	无
R_LINETAB	218	9	***ES = B1***； ***CU = B3***； ***SM = B4***
R_LINETAB_ESC	219	3	***ES = B1***； ***M = B1***
R_LTP_OVERRIDE	220	1	无
R_COMMENT	221	6	***OP = B1; V = B2 to B6***
R_TP_OVERRIDE	222	1	无
R_RESERVED	224-255		保留

参数重定位位在修正请求中以两种方式进行编码，如上表中的 *rbits1* 和 *rbits2* 所注明。

第一种编码方式认为，最常用的过程调用在参数列表中只有不带空洞的常规寄存器参数。此类调用的编码只是常规寄存器（0 到 4）中参数的数量，如果常规寄存器中有一个返回值，则再加上常规寄存器 5 中参数的数量。

第二种编码方式较为复杂。通过消除一些不可能的合并，将 10 参数重定位位压缩成 9 位。该编码由三部分组成。第一部分是返回值的位对，不可修改。如果前两个参数字可以共同形成双精度参数，则第二部分为 9；否则，它是第一个字的位对与第二个字的位对之和的 3 倍。同样，第三部分是基于第三个和第四个参数字形成的。

将第二部分乘以 40，将第三部分乘以 4，然后将这三个值加在一起。

编译程序记录

编译程序记录由每个编译程序或汇编程序放在可重定位的文件中，使用这些记录可标识用来生成该文件的编译程序的版本。这些记录由链接程序复制到可执行文件中，但它们是可去除的。下面显示了编译程序记录的结构。所有字符串都包含在符号字符串表中。

编译记录的格式通过 **<compunit.h>** 中的以下结构声明来描述。

```
struct compilation_unit {
    union name_pt     name;              /* entry name */
    union name_pt     language_name;     /* language used */
    union name_pt     product_id;        /* compiler ID */
    union name_pt     version_id;        /* compiler version */
    unsigned int      reserved: 31;      /* reserved */
    unsigned int      chunk_flag: 1;     /* MPE-only */
    struct sys_clock  compile_time;      /* time file was compiled */
    struct sys_clock  source_time;       /* time file was last modified */
};
```

文件

<a.out.h>
<aouthdr.h>
<compunit.h>
<elf.h>
<filehdr.h>
<reloc.h>
<scnhdr.h>
<spacehdr.h>
<syms.h>

另请参阅

系统工具

as(1) 将汇编代码转换为机器代码
cc(1) 调用 HP-UX C 编译程序
ld(1) 调用链接编辑器

其他信息

crt0(3) 执行启动例行程序
elf(3E) 仅适用于 ELF a.out
end(3C) 符号在程序中的上次位置
magic(4) HP-UX 实现方案的幻数

nm(1)	输出对象文件的名称列表
strip(1)	从对象文件中去除符号和行号信息

名称

acct - 每个进程记账文件格式

概要

#include <sys/acct.h>

说明

作为调用 **acct()** （请参阅 *acct*(2) ）的结果而产生的文件具有以 **<sys/acct.h>** 定义的形式存在的记录，其内容包括：

```
typedef ushort comp_t;          /* "floating point":
                                13-bit fraction, 3-bit exponent */
struct  acct {
    char    ac_flag;            /* Accounting flag */
    char    ac_stat;            /* Exit status */
    uid_t   ac_uid;             /* Accounting user ID */
    gid_t   ac_gid;             /* Accounting group ID */
    dev_t   ac_tty;             /* control typewriter */
    time_t  ac_btime;           /* Beginning time */
    comp_t  ac_utime;           /* acctng user time in clock ticks */
    comp_t  ac_stime;           /* acctng system time in clock ticks */
    comp_t  ac_etime;           /* acctng elapsed time in clock ticks */
    comp_t  ac_mem;             /* memory usage in clicks */
    comp_t  ac_io;              /* chars trnsfrd by read/write */
    comp_t  ac_rw;              /* number of block reads/writes */
    char    ac_comm[8];         /* command name */
};
#define AFORK   01              /* has executed fork, but no exec */
#define ASU     02              /* used super-user privileges */
#define ACCTF   0300            /* record type: 00 = acct */
```

在 **ac_flag** 中， **AFORK** 标志由每个 **fork()** 打开并由一个 **exec()** 关闭（请参阅 *fork*(2) 和 *exec*(2) ）。 **ac_comm** 字段继承自父进程并由任意 **exec()** 重置。每次系统以时钟周期更改进程时，也添加 **ac_mem** ，当前进程大小计算如下：

```
 （数据大小）
+（文本大小）
+（共享文本的核心进程数）
+ 总和（（共享内存段大小）
      /（附加到段中的核心进程数））
```

对于具有虚拟内存、文本、数据以及共享内存的系统，大小指的是内存段所处的部分。 **ac_mem / (ac_stime + ac_utime)** 的值可以看作平均进程大小的平均值，由共享文本修改。

驻留在计算命令源文件中的 **tacct** 结构，代表了各种计算命令使用的总体记账格式：

```
/*
 * total accounting (for acct period), also for day"
 */
struct  tacct {
  uid_t          ta_uid;              /* userid */
  char           ta_name[8];          /* login name */
  float          ta_cpu[2];           /* cum. cpu time, p/np (mins) */
  float          ta_kcore[2];         /* cum kcore-minutes, p/np */
  float          ta_con[2];           /* cum. connect time, p/np, mins */
  float          ta_du;               /* cum. disk usage */
  long           ta_pc;               /* count of processes */
  unsigned short  ta_sc;              /* count of login sessions */
  unsigned short  ta_dc;              /* count of disk samples */
  short          ta_fee;              /* fee for special services */
};
```

警告

短期命令的 **ac_mem** 值几乎没有关于该命令大小的信息，因为 **ac_mem** 在进程执行不同命令（比如 Shell）时会增加。

内核内部的结构根据版本不同而可能在没有警告的情况下进行更改。应用直接依赖于那些不支持的结构。

记账文件当前以 32 位格式编写。因此，读取该文件的 64 位应用程序需要进行特殊的规定。尤其是， **acct.h** 标题声明 **ac_btime** 字段为 **int32_t** 而不是 64 位编译中的 **time_t** 。 **ac_btime** 字段应该在使用前从应用中删除 **time_t** （例如，在对日期转换函数的调用中）。在将来的版本中，记账文件格式将变为 64 位的布局。

另请参阅

acct(2)、acct(1M)、acctcom(1M)、exec(2)、fork(2)。

符合的标准

acct：SVID2、SVID3、XPG2

名称

ar - 通用归档文件格式

概要

#include <ar.h>

说明

ar 命令用于将多个文件连接成一个归档文件（请参阅 *ar*(1) ）。归档主要供链接编辑器搜索库时使用（请参阅 *ld*(1) ）。

每个归档都以归档“幻数字符串”开头。

```
#define  ARMAG   "!<arch>\n"            /* magic string */
#define  SARMAG  8                      /* length of magic string */
```

归档“幻数字符串”后接归档文件成员。每个文件成员前面是文件成员头，其格式如下所示：

```
#define  ARFMAG      "‘\n"              /* header trailer string */
#define  AR_NAME_LEN  16                /* ar_name size, includes ‘/’ */

struct ar_hdr   /* archive file member header - printable ascii */
{
   char    ar_name[16];       /* file member name - ‘/’ terminated */
   char    ar_date[12];       /* file member date - decimal */
   char    ar_uid[6];         /* file member user id - decimal */
   char    ar_gid[6];         /* file member group id - decimal */
   char    ar_mode[8];        /* file member mode - octal */
   char    ar_size[10];       /* file member size - decimal */
   char    ar_fmag[2];        /* ARFMAG - string to end header */
};
```

文件成员头中的所有信息都是可打印的 ASCII 字符。头中的数字信息存储为十进制数字（八进制的 **ar_mode** 除外）。因此，如果归档包含可打印文件，那么归档本身也是可打印的。

ar_name 字段以斜线 (/) 结尾，并且填充空格。 **ar_date** 字段是文件的修改时间，即文件插入归档的时间。可移植归档命令 **ar** 将通用格式归档从一个系统转移到另一个系统。请注意，老版本的 **ar** 并不使用通用归档格式，而且不能使用通用归档程序读写归档。

每个归档文件以一个奇字节边界开头；如有必要，将在文件之间插入一个换行符。但是，给定的文件大小反映的是不包含填充字符的文件实际大小。

对于归档文件中的空白区域，并没有明确的规定。如果有归档符号表，归档中的首个文件名的长度为 0（例如，**ar_name[0] == ’/’** 和 **ar_name[1] == ’ ’** ）。归档成员的内容取决于计算机。更多信息请参阅 *a.out*(4) 手册条目。

包含对象文件（请参阅 *a.out*(4) ）的归档可能包含一个归档符号表。链接编辑器使用（请参阅 *ld*(1) ）符号表，以确定链接编辑过程中应加载哪些归档成员。归档符号表（如果有）始终是归档的首个成员（但不列出），而且 **ar**

将自动创建和（或）更新它。

如果归档内成员的文件名超过 15 个字节，归档将包含一个额外的特殊成员，以存储长文件名字符串表。如果 **ar_name[0] == '/'** 而且 **ar_name[1] == '/'** ，那么特殊字符串表成员的文件名长度为 0。

如果有特殊字符串表，那么它将位于所有非特殊归档成员之前。如果同时有符号表成员和字符串表成员，那么符号表成员始终在字符串表成员之前。

字符串表中的每个条目都后接一个斜线和一个换行符。表的编移量以 0 起始。如果归档成员名超过 15 个字节，成员头中的 **ar_name** 条目将不包含成员名，相反，它将包含以斜线开头的字符串表偏移量。

例如，成员名 **thisverylongfilename.o** 的 **ar_name** 字段包含 **/0** 。此值表示在字符串表中的偏移量。成员名 **yetanotherlongfilename.o** 的 **ar_name** 字段包含 **/27** 。长文件名字符串表的格式如下：

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
0	t	h	i	s	i	s	a	v	e	r
10	y	l	o	n	g	f	i	l	e	n
20	a	m	e	.	o	/	\n	y	e	t
30	a	n	o	t	h	e	r	l	o	n
40	g	f	i	l	e	n	a	m	e	.
50	o	/	\n							

另请参阅

系统工具：

ar(1)　　创建归档库

ld(1)　　调用链接编辑器

其他：

a.out(4)　　汇编程序、编译程序和链接程序输出

magic(4)　　HP-UX 实现幻数

ranlib(1)　　重新生成归档符号表

strip(1)　　从对象文件中去除符号和行号信息

警告

strip 从归档中移除归档符号列表（请参阅 *strip*(1) ）。在归档与 **ld** 链接编辑器一起使用之前，必须使用 **ar** 命令的 **-ts** 选项或 *ranlib*(1) 命令还原归档符号表。

名称

audeventstab - 定义和描述审计系统事件

说明

/usr/audit/audeventstab 文件列出审计事件编号、相应的助记符名称以及对每个事件的简要解释。允许空行和注释（以 **#** 字符开头）。此文件中的非注释、非空行包含三个部分：

event 十进制审计事件编号：用空白字符分隔的单个字段。

name 相应的助记符名称：用空白字符分隔的单个字段。

explanation 行的其余部分，位于 **#** 字符之后。

对于内核生成的审计事件，事件编号与内核内部系统调用编号相匹配，事件名称为系统调用名称。对于自审计程序发出的事件，名称是定义在 **<sys/audit.h>** 之中的宏。

举例

通过去除注释并忽略空行，从文件中提取一个事件编号和名称的列表：

```
tab='   '
sed < /usr/audit/audeventstab -e 's/#.*//' -e "/^[ $tab]*$/d"
```

作者

audeventstab 由 HP 开发。

文件

/usr/audit/audeventstab

另请参阅

audisp(1M)、audevent(1M)。

名称

audit - 审计的文件格式和其他信息

概要

#include <sys/audit.h>

说明

当用户进行安全相关的系统调用时将生成审计记录，调用 **audwrite()** （请参阅 *audwrite*(2) ）的自我审计也生成审计记录。只有超级用户才能使用审计系统。

审计记录由记录头和记录正文组成。记录头由时间、进程 ID、错误、事件类型和记录正文长度组成。时间是指完成审计事件（无论成功或失败）所花费的时间，进程 ID 是所审计的进程的 ID，事件类型字段标识审计活动的类型，长度表示以字节为单位的记录正文长度。记录头的格式是在 **<sys/audit.h>** 中定义的，如下所示：

```
struct audit_hdr {
    u_long ah_time;          /* date/time (tv_sec of timeeval) */
    pid_t ah_pid;            /* process ID */
    u_short ah_error;        /* success/failure */
    u_short ah_event;        /* event being audited */
    u_short ah_len;          /* length of variant part */
};
```

在审计记录中，记录内容的长度是可变的，它包含审计活动的更多相关信息。对于系统调用生成的记录，记录内容包含系统调用的参数；对于自我审计进程生成的记录，记录内容由事件的高级描述信息组成（请参阅 *audwrite*(2) ）。

审计文件中的记录被压缩，以减小文件所占用的空间。首次审计一个进程时，将向审计文件写入一个 *pid* 标识记录 (PIR)，这个记录包含的信息在整个进程周期都保持不变。这包括父进程 ID、审计 ID、实际用户 ID、实际组 ID、有效用户 ID、有效组 ID 和终端 ID (tty)。对于每个审记文件的每个进程，只输入一次 PIR， PIR 也是在 **<sys/audit.h>** 中定义的，其格式如下所示：

```
struct pir_body {            /* pir-related info */
    pid_t   ppid;            /* parent process ID */
    int32_t aid;             /* audit ID */
    uid_t   ruid;            /* user_ID */
    gid_t   rgid;            /* group ID */
    uid_t   euid;            /* effective user_ID */
    gid_t   egid;            /* effective group_ID */
    dev_t   tty;             /* tty number */
};
```

audisp 分析和显示审计文件中累积的信息（请参阅 *audisp*(1M) ）。

无论何时开启审计，都需要一个“当前”审计文件，并建议使用一个“后续”审计文件（用于备份）（请参阅

audsys(1M) 和 *audomon*(1M)）。当“当前”审计文件已满而且“后续”审计文件可用时，审计系统将自动切换文件。

作者

audit 由 HP 开发。

另请参阅

audsys(1M)、 audevent(1M)、 audisp(1M)、 audomon(1M)、 audwrite(2)、 getevent(2)、 setevent(2)。

名称

authcap - 可信系统的安全数据库

概要

/tcb/files/auth/*

/tcb/files/auth/system/*

说明

在文件系统中，所有与安全相关的数据库都是以 ASCII 格式存储的。联机帮助页第 3 节中介绍的支持例行程序可以将 ASCII 格式转换为二进制结构。本联机帮助页介绍这些数据库的格式，并介绍如何将它们转换为数据结构。

层次结构

完整的数据库位于两个目录层次： **/tcb/files/auth/*** 和 **/tcb/files** 。第一个目录层次包含受保护口令数据库，而且包含一些目录名为单字母名称的子目录，每个单字母名称都是用户名的首个字母。这些子目录内包含常规文件，每个常规文件都包含一个 *authcap*(4) 格式文件，而这个格式文件则包含特定用户的受保护口令条目。因此，所有以 **x** 开头的用户名都在 **/tcb/files/auth/x** 目录下有一个包含了相应验证和身份信息的文件。

/tcb/files/auth/system 和 **/tcb/files** 内的目录包含在系统级使用的信息。全局系统设置位于目录 **/tcb/files/auth/system** 。终端和设备分配文件位于目录 **/tcb/files** 。

以下数据库文件位于目录 **system** ：

default	缺省控制

以下数据库位于目录 **/tcb/files** ：

ttys	终端控制
devassign	设备分配

文件格式

所有数据文件（ /tcb/files/auth/system 目录下的文件和 **/tcb/files** 目录下的文件）都具有相同的格式。每个文件都包含一个虚拟行，可以通过在行末尾插入字符（最后一行除外）将它分为物理上的多个行。例如，下面行：

```
smk:u_name=smk:u_id#16:u_pwd=a78/a1.eitfn6:chkent:
```

可以分为：

```
smk:u_name=smk:u_id#16:\
   :u_pwd=a78/a1.eitfn6:\
   :chkent:
```

请注意，必须用 **:** 分隔符来分隔各个功能部分。多行条目需要每行末尾和每个连续行的开头都有一个 **:** 分隔符。连续行用制表符缩进。多个条目之间用换行符分隔，而且换行符前面没有续行符。

```
daa:u_name=daa:u_id#75:u_maxtries#9:chkent:
smk:u_name=smk:u_id#76:u_maxtries#5:chkent:
```

行格式

行格式如下所示：

name:cap1:cap2:cap3:...:capn:chkent:

条目是通过名称来引用的。条目中的名称部分以 **:** 字符终止。

条目的末尾是一个 *chkent* 字段。它用于条目的完整性检查。 authcap 例行程序拒绝所有不包含 *chkent* 终止符的条目。

每个条目包含零个或多个功能部分，每个功能部分以 **:** 字符终止。每个功能部分都有唯一的名称。数字功能部分格式如下：

*id***#***num*

num 是一个十进制数或一个（以 0 开头的）八进制数。布尔功能部分格式如下：

id

或者

*ID***@**

第一种形式表示不具有这个功能，第二种形式表示具有这个功能。字符串功能部分格式如下：

id=string

这里字符串可以是零个字符或多个字符。 **** 和 **:** 字符分别转义为 **** 和 **\:** 。

文件锁

所有数据库都使用锁定文件，锁定文件表示文件正在被重写。有时，系统崩溃后锁定文件将保留，这就必须手动删除锁定文件。在数据库文件名后追加 **-t** 将形成锁定文件。

字段/标志

程序可以将所有数据库转换为结构。数据结构由两个子结构组成，每个子结构都包含与数据库条目中的字段相对应的成员。 *field* 结构包含一个字段值（例如，一个数字、布尔标志、目录字符串或掩码），而标志值（一位）表明具有或缺少条目中的字段。

作者

authcap 由 HP 开发。

另请参阅

getdvagent(3)、 getprdfent(3)、 getprpwent(3)、 getprtcent(3)、 default(4)、 devassign(4)、 prpwd(4)、 ttys(4)。

名称

bootconf - 引导设备配置表

说明

文件 **/stand/bootfonf** 包含系统的引导设备或 *lif* 卷的地址及磁盘布局类型。 **Software Distributor** 和 HP-UX 内核控制脚本（文件集 **OS-Core.KERN-RUN** ）使用这个文件来确定在何处以及如何更新初始引导加载程序。通常情况下，内核的 **checkinstall** 脚本查询系统硬件，并创建这个文件。在极少情况下，如果不能自动确定系统配置，或者需要自动更新附加和（或）备用引导设备，管理员应手动编辑 **/stand/bootconf** 文件。

每个引导设备的文件都包含一行。每行都包含用空格分隔的字段，各字段的顺序如下：

disk type 这个标志说明磁盘上的文件系统布局。它必须是以下标志之一：

l 表示根磁盘是 LVM 或 VERITAS 卷管理器 (VxVM) 格式的。如果使用 LVM 或者 VxVM 镜像，那么每个“镜像”必须有自己的行。

p 表示根磁盘具有 800 系列硬分区而且引导卷是第 6 扇区。

w 表示根磁盘是 9.X 版本的 700 系列“整磁盘”格式，无分区，引导和交换空间保留在文件系统之外。

device file 表示访问引导区域所在物理设备的设备专用文件的绝对路径。对于 LVM 根磁盘，设备专用文件是 **vgdisplay -v** 命令返回的物理卷。对于 800 系列硬分区，设备专用文件指向磁盘的第六扇区。对于 Series 700-style “整磁盘”，设备文件引用整个磁盘。

允许使用空白行。所有以 **#** 开头的行都被视为注释。

诊断信息

如果 **bootconf** 文件正确，软件分发日志文件 **/var/adm/sw/swagent.log** 将包含 **OS-Core.KERN-RUN** 文件集下的诊断消息。大多数消息都是自述性的；少数消息需要额外的解释：

... is either empty or improperly formatted...

如果没有其他有关 **bootconf** 的信息，说明文件可能是空的。或者文件的格式不正确，其他信息将解释问题的所在。

device file **... does not contain a valid boot LIF ...**

指定的设备文件没有指向 *lif* （它包含文件 **HPUX** ）所在的磁盘。

... has an invalid character in the flag field...

行的第一个字段含有除 **#** 、 **l** 、 **p** 或 **w** 以外的其他字符。

... contains contradictory boot LIF types...

对于版本 10.0， **/stand/bootconf** 中的引导区域必须在相同类型的磁盘布局上。

... has unrecognized extra characters...

device file 规范后有字符。

举例

引导区域在 LVM 根磁盘上:

```
# Boot Device configuration file
# This file contains information regarding the location
# of the boot LIF. It is used by the KERN-RUN fileset to
# update the boot kernel.
l /dev/dsk/c2t7d0
```

系统在根磁盘上有一个 LVM 镜像（设备文件表示系统运行的 9.0 版本正准备升级到 10.0）:

```
# Boot Device configuration file
# This file contains information regarding the location
# of the boot LIF. It is used by the KERN-RUN fileset to
# update the boot kernel.
l /dev/dsk/c1d0s2
l /dev/dsk/c4d0s2
l /dev/dsk/c5d0s2
```

引导区域在硬分区的磁盘上:

```
# Boot Device configuration file
# This File contains information regarding the location
# of the boot LIF. It is used by the KERN-RUN fileset to
# update the boot kernel.
p /dev/dsk/0s0
```

引导区域在整磁盘布局上:

```
# Boot Device configuration file
# This File contains information regarding the location
# of the boot LIF. It is used by the KERN-RUN fileset to
# update the boot kernel.
w /dev/dsk/6s0
```

警告

文件中的所有引导设备必须具有相同的磁盘布局。

作者

bootconf 由 HP 开发。

文件

/stand/bootconf

另请参阅

mediainit(1)、 hpux(1M)、 mkboot(1M)、 vgdisplay(1M)、 lif(4)。

Software Distributor 文档资料。

名称

cdnode - CDFS 的 cd 节点格式

概要

#include <sys/types.h>

#include <sys/cdnode.h>

说明

本条目描述了用于 CDFS 文件系统的 cd 节点结构及相关概念。

CDFS 文件系统中没有一个称为 i 节点的独立实体的概念。通常位于 HFS 的 i 节点中的信息保存在 **cdnode** 数据结构中。然而，cd 节点数据结构不保存在物理介质上，而是只保存在核心内存空间。 cd 节点信息用于唯一标识一个文件。

保存在 cd 节点结构中的信息来自于 CDFS 文件系统中的其他两个数据结构：

1. 用于文件或目录的目录记录，和
2. 如果存在，则用于文件或目录的扩展属性记录 (XAR)。

因为通常只有少数文件具有与之关联的 XAR，所以 cd 节点信息大多数只是由文件目录记录提供的属性组成。

由于 cd 节点保存在核心内存中，因此用户不可直接访问。 **stat()** 系统调用试图将包含在给定文件的 cd 节点中的信息，映射成标准的 stat 结构（请参阅 *stat*(2) ）。但是，由于 cd 节点中包含的信息在 stat 结构中没有相应的字段，信息不能被映射，因此也不能访问。尚未提供访问完整的 cd 节点结构的方法。

文件

/usr/include/sys/cdnode.h

/usr/include/sys/cdfsdir.h

另请参阅

stat(2)、cdrom(4)、cdfsdir(4)。

名称

cdrom - CD-ROM 背景信息

说明

本联机帮助页介绍现有的 CD-ROM 标准、术语、数据布局和支持级别等一般信息。有关详细信息，请参阅另请参阅中列出的标准文档。

当前的 HP-UX 版本并不支持在此讨论的所有主题。关于当前版本的详细内容，请参考相关内容一节。

标准格式

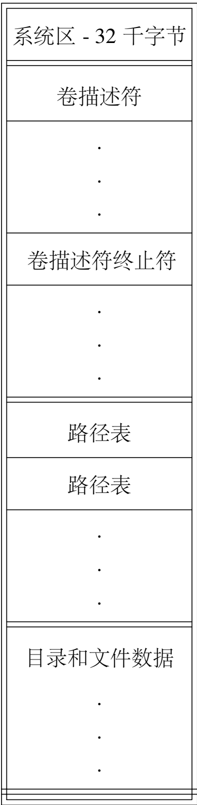
当前，为 CD-ROM 定义了两种标准格式。

High Sierra Group (HSG) 标准是由 CD-ROM Ad Hoc Advisory Committee 制定的，标题为《The Working Paper for Information Processing – Volume and File Structure of Compact Read Only Optical Discs for Information Interchange》的文档阐述了该标准。此文档可以从国家信息标准组织（National Information Standards Organization，NISO）获取。

第二个标准是从 HSG 标准发展而来的，它由 ISO（International Organization for Standardization，国际标准化组织）制定。标题为《Information Processing – Volume and File Structure of CD-ROM for Information Interchange》的文档阐述了该标准，其参考编号为 ISO 9660:1988 (E)。

数据布局

CD-ROM 上的数据布局表示如下:



CD-ROM 上的数据一般包括四个部分（在上表中用双横线指示）：只有前两个部分必须按如上所示的顺序出现。

“系统区”由介质上前 16 个 2048 字节的块组成。 CD-ROM 的两种标准都没有对系统区的内容作出规定； CD-ROM 的创建者可以在这个区域存放与 CD-ROM 的目标系统相关的信息。

“卷描述符”一般包含一个主卷描述符和零或多个辅助卷描述符。每一个卷描述符的长度都是 2048 字节，它描述 CD-ROM 上目录层次结构的属性和结构。卷描述符列表由一个或多个 “卷描述符终止符”来结束。卷描述符终止符的长度也是 2048 字节，它仅仅表示卷描述符部分的结束。

“路径表”包含 CD-ROM 上所有目录层次结构的所有路径表。路径表不一定在 CD-ROM 数据的该部分，它可以分布在 “目录和文件数据”部分（下面将描述），从而使寻道时间最小化。

“目录和文件数据”包含 CD-ROM 上所有目录层次结构的数据，如前所述，它还可能因为偶尔包含路径表而变得不连续。

卷和目录层次结构

“卷”是单个物理 CD-ROM。 “目录层次结构”是写入到卷上的分层文件系统。一个卷上可能有多个目录层次结构，一个目录层次结构也可能跨越多个卷。一个 “卷描述符”描述卷上的一个目录层次结构。

同一个卷上的各个目录层次结构可以完全相互独立，每个目录层次结构可以完全唯一地进行定义，并且不与文件系统关联。目录层次结构之间还可以通过数据共享来互相关联。

“卷集”是由一个或多个卷组成的集合，它被视为一个单元。卷集中的每个连续卷都更新或扩充它前面卷上的数据。因此，卷集中的最后一个卷总是描述卷集最新的目录层次结构的卷。将为卷集中的每个卷指定一个称为 “卷序列号”的唯一且递增的值。在更新大型多卷数据库而不更改整个集时，卷集是很有用的。

卷描述符

卷上的每个目录层次结构由一个 “卷描述符”进行描述。卷描述符有多个类型，但最重要的两个类型是 “主卷描述符”和 “辅助卷描述符”。它们的内容几乎是相同的，但是它们的用途不同。

主卷描述符描述卷上的主目录层次结构。如果卷上还有其他目录层次结构，或者同一目录层次结构具有不同的查看方式，则应该使用辅助卷描述符来描述。对于卷集，每个卷上的主卷描述符描述该卷和卷集中该卷之前所有卷的主目录层次结构。

卷描述符包含以下信息:

标准 ID（标识卷的格式）;
系统 ID;
卷 ID;
卷的大小;
卷集大小;
卷序列号;
逻辑块大小;
路径表大小;
路径表指针;
根目录的目录记录;
卷集 ID;
出版商 ID;
数据准备者 ID;
应用程序 ID;
版权文件名称;
摘要文件名称;
文献目录文件名称（仅适用于 ISO 标准）;
卷创建日期和时间;
卷修改日期和时间;
卷到期日期和时间;
卷有效日期和时间;
应用程序使用区。

路径表

“路径表”定义卷内的一个目录层次结构。每个路径表都包含层次结构中每一个目录的记录。每个记录保存了目录名称、与目录相关的扩展属性记录的长度、目录的启始块的逻辑块编号和目录的父目录编号。（路径表中所有目录都有一个编号，这个编号是根据它们在路径表中的出现顺序排列的）。

路径表有两种类型。一类是 **L** 型路径表，在这类路径中，路径表记录的所有值都是以最低有效字节在先的方式存储的。另一类是 **M** 型路径表，在这种类型中，所有值都是以最高有效字节在先的方式存储的。 CD-ROM 的两种标准都各需要一个 L 型路径表和一个 M 型路径表。 ISO 标准允许每类路径表有一个附加的可选副本， HSG 标准允许每个类型最多有三个附加的可选副本。对于冗余和寻道时间最小化，路径表的附加副本是很用的。

扩展属性记录

“扩展属性记录”（缩写为 **XAR** ）是一个数据结构，它指定与 XAR 相关的文件和目录的附加信息。 XAR 包含以下信息：

所有者 ID；
组 ID；
权限；
创建日期和时间；
修改日期和时间；
到期日期和时间；
有效日期和时间；
记录信息；
应用程序使用区；

如果记录了 XAR，那么 XAR 存放在文件或目录的第一个块的开头部分。文件或目录的实际数据存储于 XAR 所在块之后的各个块中。

如果可能，可以通过 **stat()** 系统调用（请参阅 *stat*(2) ）将 XAR 信息映射到 stat 结构中。但是，很多项目不能被很好地映射，这是因为 stat 结构缺乏相应的字段来存放 XAR 提供的信息。为了保持向后兼容， **stat()** 将丢弃这些信息。可以使用 **fsctl()** 系统调用来获取特定文件或目录的 XAR（请参阅 *fsctl*(2) ）。

交错

出于性能的考虑，文件的数据可能交错地记录在卷上。通过将文件分成名为 “文件单元”的文件片断，可以实现这一点（逻辑块中的）文件单元的大小称为 “文件单元大小”。写入一个文件单元，跳过一个或多个块，再写入另一个块，再跳过一个或多个块，这样反复进行，直到记录完整个文件，这样就将交错文件记录到了卷上。文件单元之间被跳过的块的数量称为 “交错间隔大小”。其他文件可以使用用于弥补交错间隔的块。

每个文件的目录记录都记录了文件单元和交错间隔的大小。因此，各个文件的文件单元和交错间隔的大小可能各不相同，但是在同一文件内，它们是不变的（除非文件写入到 节 – 中，请参阅下面的内容）。

目录不能是交错的。

文件节

为了在文件之间共享数据，一个文件可能被分成多个 “文件节”。一个文件的各个文件节的大小可能是不相同的。

每个文件节都被视为一个单独的文件，它们都有自己的目录记录。这就意味着每个文件节都有自己的大小、自己的 XAR 以及自己唯一的文件单元大小和交错间隔大小。但是，同一文件的各个文件节必须共享相同的文件名。在文件中，文件节的顺序由各节的目录记录顺序确定。目录记录中的一个位确定一个记录是否为文件的最后一个记录。

一个文件节可以在单个文件中出现一次或多次，也可以在不同文件中出现多次。一个卷中的文件节还可以被卷集中的后续卷使用（更新就是通过这种方式来完成的）。

每个文件节都可以有自己的 XAR。但是，如果文件的最后一个文件节没有相关的 XAR，那么整个文件都将被视为没有 XAR。这样做是为了更易发现文件的更新。

目录必须始终是单个节。

实现级别和交换级别

CD-ROM 标准定义了两级实现级别和三级交换级别。 “实现级别”为支持 CD-ROM 的接收系统提供一种指定支持级别的方式。实现级别是:

Level 1　允许系统忽略辅助卷描述符以及与辅助描述相关的路径表和所有目录和文件数据。

Level 2　没有应用任何限制。

在所有情况下，接收系统都必须符合在 ISO 标准的第 10 节中指定的系统要求（HSG 标准没有相对应的要求）。

“交换级别”提供一种方式以指定数据结构和 CD-ROM 上的复杂度。这些级别是:

Level 1　每个文件由一个文件节组成。文件名不超过 8 个字符，文件扩展名不超过 3 个字符。目录名不超过 8 个字符。

Level 2　每个文件由一个文件节组成。

Level 3　没有应用任何限制。

注释

可以使用 PFS（可移植文件系统）工具来支持其他 CD-ROM 标准。有关详细信息，请参阅 *pfs*(4) 。

相关内容

HP-UX 仅支持主卷描述符。挂接卷后，HP-UX 挂接由其查找到的主卷描述符所描述的目录层次结构。 HP-UX 识别并忽略辅助卷描述符及其相关的目录层次结构。

HP-UX 不支持跨多个卷的目录层次结构。

HP-UX 不支持由多个卷组成的卷集。

HP-UX 忽略路径表。 HP-UX 使用 HFS 文件系统中的标准路径名查找方案。这样做可在已挂接的 CDFS 文件系统上挂接其他可挂接的文件系统。此外，由于 HP-UX 维护 CDFS 文件的 cdnode 缓存（请参阅 *cdnode*(4) ），从

而使路径表提供的额外性能优势降到最小。

HP-UX 不支持多文件节。每个文件必须记录在一个文件节中。

HP-UX 支持级别 1 实现和级别 2 交换。

另请参阅

fsctl(2)、 stat(2)、 cdnode(4)、 pfs(4)。

《Information Processing - Volume and File Structure of CD-ROM for Information Interchange》， Ref. No. ISO 9660 : 1988 (E)。

《The Working Paper for Information Processing – Volume and File Structure of Compact Read Only Optical Discs for Information Interchange》， National Information Standards Organization [Z39]。

名称

charmap - localedef 脚本的符号转换文件

概要

localedef -f *charmap* **locale_name**

说明

用 **-f** 选项调用 **localedef** 命令，将导致 语言环境描述文件中的符号名称被转换为 *charmap* 文件中给定的编码（请参阅 *localedef*(1M)）。建议完全用符号名称来编写语言环境描述文件。

charmap 文件有两部分：声明部分和字符定义部分。

声明部分

声明部分可以位于字符定义部分之前。

每个部分由符号（包括两端的尖括号）、一个或多个空格（或制表符、空格字符等）以及符号的值构成。

多字节字符代码集需要某些声明。对于单字节代码集，这些声明都是可选的。

下面列出了可能的声明：

<code_set_name> *value*

用于声明已定义了 charmap 文件的编码字符集的名称。多字节字符编码集需要这个关键字。对于 HP15 编码方案，名称需要包含 **HP15** 。对于 EUC 编码方案，名称需要包含 **EUC** 。

<cswidth> *value*

用于声明已定义了 charmap 的文件的编码字符集的 cswidth 参数（请参阅 *eucset*(1)）。

<mb_cur_max> *value*

声明多字节字符的最大字节数。如果没有声明，则缺省值为 1。对于多字节字符代码集，必须指定此关键字。

<mb_cur_min> *value*

为编码字符集声明字符的最小字节数。此值必须小于或等于 **<mb_cur_max>** 。如果没有声明，缺省值等于 **<mb_cur_max>** 。

<escape_char> *value*

声明转义符；转义符用于转义具有特殊含义的字符。如果没有声明，缺省转义符是反斜杠 (\) 。

<comment_char> *value*

声明注释字符；注释字符用于开始注释，它应放在 *charmap* 文件的第一列。如果没有声明，缺省注释字符是 **#** 字符。

字符定义部分

字符集映射定义部分恰好位于包含字符串 **CHARMAP** 的标识行与包含字符串 **END CHARMAP** 的结尾行之间（空行和以注释字符开头的行被忽略）。

字符定义有两种形式。

第一种形式定义单字符及其编码:

<*symbolic_name***>** *encoding*

symbolic_name 是取自于 XPG 规范可移植字符集中的一个或多个可见字符，它会用尖括号括起来。如果要在名称中使用尖括号、转义字符或注释字符等元字符，必须对它们进行转义。可以为相同编码指定两个或多个符号名称。

encoding 是一个字符常量，其形式是以下三种形式中的任意一种:

decimal	转义字符后接字母 **d**，再后接一至三个十进制数字。
octal	转义字符后接一至三个八进制数字。
hexadecimal	转义字符后接 **x**，再后接两个十六进制数字。

多字节字符通过连接多个字符常量来表示。多字节字符编码中使用的所有常量必须具有相同的形式。

第二种形式定义一个字符范围，这个字符范围涵盖从第一个符号名称到第二个符号名称的所有字符:

<*symbolic_name***>**... **<***symbolic_name***>** *encoding*

符号名称必须以一个或多个非数字字符开头，后接由一个或多个十进制数字构成的整数。第二个符号名称的整数部分必须大于第一个符号名称的整数部分。范围被解释为一系列符号名称，列表中的符号名称具有相同的字符部分，并且从第一个符号名称到最后一个符号名称，其整数部分的值都是连续的。为符号名称分配以给定编码开始的多个连续编码。

例如，字符定义行

<C4>...<C6> \d129

它等效于:

<C4> \d129
<C5> \d130
<C6> \d131

举例

相关示例请参阅 **/usr/lib/nls/loc/charmaps** 目录中的任意文件。

另请参阅

localedef(1M)、localedef(4)。

符合的标准

localedef POSIX.2、 XPG4。

名称

core - 核心映像文件格式

说明

当接收到某些信号时， HP-UX 系统会输出一个包含已终止进程的核心映像的文件（有关原因列表，请参阅 *signal*(5)）。最常见的原因是内存违反、非法指令、浮点异常、总线错误和用户生成的退出信号。核心映像文件称为 **core**，该文件将被输出到进程的工作目录中（条件是普通访问控制允许此操作）。有效用户 ID 与实际用户 ID 不同的进程不生成核心映像。

该文件包含充足的信息用以决定终止时进程的运行情况。核心文件内容包括表示各不同进程段的对象。每个对象之前加有一个 **corehead** 数据结构，并且每个 **corehead** 数据结构描述跟随其后的相应对象。该结构在 **<sys/core.h>** 中定义，并包含下列成员：

```
int       type;
space_t   space;
caddr_t   addr;
size_t    len;
```

space 和 *addr* 成员指定进程中的虚拟内存地址，该地址就是所描述的对象开始的地址。 *len* 成员是对象的长度，以字节为单位。

type 的下列可能值在 **<sys/core.h>** 中定义：

CORE_DATA 在核心映像创建时的进程数据。包括已初始化的数据、未初始化的数据和核心映像生成时的堆。

CORE_EXEC 一个包含 exec 数据结构的编译器相关数据结构、可执行文件的幻数和命令（请参阅 **<sys/core.h>** 中的 **proc_exec** 结构声明）。

CORE_FORMAT 生成的核心格式的版本号。此编号随每一个其核心格式自身已更改的 HP-UX 版本而改变。然而，它不一定随每个 HP-UX 版本而改变。因此核心读取工具可以轻松使用 **CORE_FORMAT** 来决定它们是否与一个给定的核心映像兼容。此类型由一个四字节的二进制整数表示。

CORE_KERNEL 核心映像生成时，与内核相关联的以 null 结尾的版本字符串。

CORE_PROC 一个包含每个进程信息如硬件寄存器内容的体系结构相关数据结构。请参阅 **<sys/core.h>** 中的 **proc_info** 结构声明。

CORE_STACK 核心映像创建时的进程堆栈内容。

在 **core** 映像文件中转储的对象不按照任何特定的顺序排列。使用 **corehead** 信息决定紧随其后的对象类型。

另请参阅

adb(1)、 cdb(1)、 xdb(1)、 setuid(2)、 crt0(3)、 end(3C)、 signal(5)。

名称

cpio - cpio 归档格式

说明

当 **cpio** 的 **-c** 选项未使用时（请参阅 *cpio*(1)）， *header* 结构如下：

```
struct {
   short   c_magic,
           c_dev;
   ushort  c_ino,
           c_mode,
           c_uid,
           c_gid;
   short   c_nlink,
           c_rdev,
           c_mtime[2],
           c_namesize,
           c_filesize[2];
   char    c_name[c_namesize rounded to word];
} Hdr;
```

当 **cpio -c** 选项在使用时， *header* 信息描述如下：

```
sscanf(Chdr,"%6ho%6ho%6ho%6ho%6ho%6ho%6ho%6ho%11lo%6ho%11lo",
      &Hdr.c_magic,&Hdr.c_dev,&Hdr.c_ino,&Hdr.c_mode,
      &Hdr.c_uid,&Hdr.c_gid,&Hdr.c_nlink,&Hdr.c_rdev,
      &Longtime,&Hdr.c_namesize,&Longfile);
```

Longtime 和 *Longfile* 分别等同于 **Hdr.c_mtime** 和 **Hdr.c_filesize** 。每个文件的内容和描述文件的其他项目一起记录。每个 **c_magic** 实例包含常量 070707（八进制）。项目 **c_dev** 通过 **c_mtime** 具有了在 *stat*(2) 中所述的含义。以空字符结尾的路径名 **c_name** 的长度，包括空字符字节在内，由 **c_namesize** 给定。

archive 的最后一个记录始终包含了名称 **TRAILER!!!** 。目录和页脚与等于零的 **c_filesize** 一起记录。

c_dev 和 **c_ino** 的值并非总是与 **stat()** 的结果相对应，但是他们的值足以表明归档中的两个文件是否相互链接。

当一个设备专用文件由 HP-UX **cpio** （使用 **-x** 选项）来归档时， **c_rdev** 包含了一个幻常量，依靠实现写操作来确定值。 **H_rdev** 将设备文件标志为一个 HP-UX 32 位设备说明符，并且 **c_filesize** 包含了 32 位设备说明符（请参阅 *stat*(2)）。如果 **-x** 选项未使用，则专用文件不可归档或恢复。非 HPUX 设备专用文件从不进行恢复。

另请参阅

cpio(1)、 find(1)、 stat(2)。

符合的标准

cpio：XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1

名称

default - 信任系统的系统缺省数据库文件

概要

/tcb/files/auth/system/default

说明

系统缺省数据库的独特之处在于它定义了信任系统的系统级全局参数。其目的是为用户和设备提供全局范围的值，对于一些相同的值，管理员无须将它们分别复制到各个用户数据库和设备数据库之中。除了更易指定全局值，它还使得必要情况下的全局系统修改更为容易。

系统缺省数据库由四种类型的值组成：

系统级参数	这些值是一些参数，信任系统的任何其他数据库并不对这些参数进行相应的说明。如果系统级的参数没有在缺省数据库中指定，参数将被视为没有定义。
用户参数	这些参数一般是在受保护口令数据库文件中指定的。
终端控制参数	这些参数一般是在终端控制数据库文件中指定的。
设备分配参数	这些参数一般是在设备分配数据库中指定的。

可以为受保护口令数据库、终端控制数据库、设备分配数据库中查找到的条目指定系统缺省参数。从这些数据库中检索到特定条目以后，将向调用者返回一个名为 *ufld* 的结构，这个结构包含了所有显式指定的值。还提供第二个名为 *sfld* 的结构，这个结构定义系统缺省数据库提供的值。这些结构中的每一个都有一个名为 *uflg* 标志结构，第二个结构有一个名为 *sflg* 的标志结构，这两个标志结构说明结构中的哪些字段已经指定并可使用。程序将优先使用用户特定值或设备特定值（如果已经提供）。否则，程序将选择使用系统缺省值（如果已经指定）。如果都没有指定，程序将提供一个合理的值，或者异常终止。

对于受保护口令数据库、终端控制数据库和设备分配数据库提供的特定字段描述，请参阅另请参阅中列出的数据库的联机帮助页。下列字段是系统缺省数据库所特有的字段，不能在任何其他系统数据库中指定这些字段。

d_name	此名称被设置为字符串“default”。
d_boot_authenticate	这个标志字段表示引导计算机是否要求引导验证。如果要求验证，系统程序 *init*(1M) 将在完成系统引导之前进行验证。

举例

以下是一个典型的系统缺省数据库示例。有关文件和行格式描述，请参阅 *authcap*(4) 。

```
default:\
    :d_name=default:\
    :d_boot_authenticate@:\
    :u_pwd=*:\
    :u_minchg#0:u_maxlen#10:u_exp#15724800:u_life#31449600:\
    :u_pickpw@:u_genpwd@:u_restrict@:u_nullpw@:\
    :u_genchars@:u_genletters@:\
```

```
:u_maxtries#5:u_lock:\
:t_logdelay#2:t_maxtries#10:\
:chkent:
```

系统缺省数据库定义受到支持的四种不同类型的值。首先，定义了仅用于在系统级分配的值。不启用系统启动引导验证。如果当前系统时间距口令过期时间不足 604800 秒（可转换为 60*60*24*7 或 7 天），登录程序将发出口令过期警告信息。

系统缺省数据库还定义了很多受保护口令数据库缺省值。以 **u_** 开头的字段对应于受保护口令字段。与此类似，以前缀 **t_** 开始的字段是终端控制数据库字段。如果相应数据库没有提供用户特定值或设备特定值，那么可以使用字段类型来提供系统级的缺省值。有关适用字段的完整描述，请参阅另请参阅中列出的数据库的相关联机帮助页。

文件

/tcb/files/auth/system/default 信任系统的系统缺省数据库文件，请参阅 *authcap*(4)

/tcb/files/auth/*/* 受保护口令数据库，请参阅 *prpwd*(4)

/tcb/files/ttys 终端控制数据库文件，请参阅 *ttys*(4)

/tcb/files/devassign 设备分配数据库文件，请参阅 *devassign*(4)

作者

default 由 HP 开发。

另请参阅

getprdfent(3)、authcap(4)、devassign(4)、prpwd(4)、ttys(4)。

名称

devassign - 信任系统的设备分配数据库文件

概要

/tcb/files/devassign

说明

系统支持包含本地登录终端条目的单个设备分配数据库。

终端控制数据库的格式与其他信任系统验证数据库文件相同。关于文件格式的详细信息，请参阅 *authcap* (4)。该文件包含关键字字段标识符及那些字段的值。支持的关键字标识符及其使用如下：

v_devs 本字段标识符指定了分隔别名列表的逗号，而别名指的是该条目定义的同一设备。使用本字段避免了重复所有设备别名设备分配数据库的需要。

v_type 本字段指定了由该条目描述的设备。支持的设备类型有：

terminal 该设备作为本地登录终端设备分配。

v_users 如果指定了该字段，则该字段包含一个逗号分隔的用户名列表，该列表中的用户可以使用该设备进行登录或数据的导入/导出。如果没有该列表，则允许所有用户使用该设备。如果有该列表，则通过 *login* 程序搜索匹配来决定是否允许该用户使用该设备。

举例

以下是设备分配数据库条目的例子，该数据库条目针对指定为登录设备的终端设备：

```
tty0:v_devs=/dev/tty0:\
    :v_type=terminal:\
    :chkent:
```

警告

不应将远程终端 (ptys) 添加到 **devassign** 或 **ttys** 数据库。作为 ptys 登录的设备名称格式有：

ptym/*
pts/*
pty/*
pty[*x*][*y*] 其中 *x* 是一个字母，而 *y* 是一个十六进制数
tty[*x*][*y*] 其中 *x* 是一个字母，而 *y* 是一个十六进制数
telnet/*

作者

devassign 由 HP 开发。

另请参阅

login(1)、 getdvagent(3)、ttys(4)、authcap(4)、default(4)。

名称

dialups、d_passwd - 拨号安全控制

说明

dialups 和 **d_passwd** 用于控制 **login** （请参阅 *login*(1)）的拨号安全特性。如果有 **/etc/dialups** 文件，每行的第一个单词在登录执行时会与行的名称作比较，包括 **/dev/** ，通过 **ttyname()** 返回（请参阅 *ttyname*(3C)）。如果登录在 **dialups** 找到的一行发生，则调用拨号安全。空格或制表符后面的任何内容都被忽略。

当调用拨号安全时， **login** 请求一个附加口令，并将这个口令与 **/etc/d_passwd** 中查找到的口令进行比较。在 **/etc/passwd** 的"program to use as shell"字段中查找到的命令名称用于选择需要使用的口令。 **d_passwd** 中的每个条目由三个用冒号分隔的字段组成。第一个字段是命令名称，它与 **passwd** 中的条目相匹配。第二个字段是拨号安全机制所使用的加密口令，用于验证登录并希望使用程序的用户。第三个字段是注释，但是需要使用第二个冒号来界定口令的结尾。空口令用相邻的两个冒号表示。如果其他条目不能够与取自于 **passwd** 的命令名称相匹配，将使用 **/usr/bin/sh** 的条目。

文件

/etc/dialups	拨入 tty 连线
/etc/d_passwd	口令

另请参阅

login(1)、 passwd(4)。

名称

dir - 短文件名 HFS 文件系统上的目录格式

概要

#include <sys/types.h>

#include <sys/dir.h>

备注

本条目描述 HFS 文件系统上的与系统 V 兼容的目录格式。提供它完全是为了向后兼容，并与需要使用系统 V 文件系统环境的应用保持兼容。它不能兼容 **<dirent.h>** 中的与之相似但更为通用的 HFS 目录格式，dirent.h 描述了一个与支持最长文件名为 255 个字符的 HFS 文件系统中使用的格式相等同的格式。

定义在 **<dirent.h>** 中的 **dirent** 结构应与 *directory*(3C) 例行程序一起使用，使之能够移植到其他行业的 UNIX 实现。

说明

目录与普通文件几乎完全相同，但是用户不能写入目录。 **<sys/dir.h>** 头文件中给定的目录条目的结构如下所示：

```
#define DIRSIZ          14
#define DIRSIZ_CONSTANT    14
#define DIR_PADSIZE       10
#define MAXNAMLEN         255
struct     direct {
   u_long  d_ino;                  /* inode number of entry */
   u_short d_reclen;               /* length of this record */
   u_short d_namlen;               /* length of string in d_name */
   char    d_name[DIRSIZ_CONSTANT];
   char    d_pad[DIR_PADSIZE];
};

/*
 * DIRSTRCTSIZ is the number of bytes in the structure
 * representing a System V-compatible (14-character
 * maximum file name length) HFS directory entry.
 */

#define DIRSTRCTSIZ  32       /* sizeof(struct direct) */
```

按照惯例，每个目录中的头两个条目是 **.** 和 **..** （“点”和“点点”）。每一个条目是目录本身。第二个条目是父目录。对于主文件系统的根目录， **..** 的含义作了修改；因为没有父目录，所以 **..** 和 **.** 的含义是相同的。

作者

dir 由 AT&T 和 HP 联合开发。

另请参阅

directory(3C)。

名称

disktab - 磁盘描述文件

概要

#include<disktab.h>

说明

disktab 是一个简单数据库，它描述磁盘几何数据。 **disktab** 中的条目由多个用冒号分隔的字段组成。每个磁盘的首个条目给定用于识别磁盘的名称，名称之间用竖线字符 (|) 分隔。

提供此文件仅仅是为了向后兼容以前版本的 HP-UX。建议不要使用它。

下面的列表列出了每个磁盘条目的标准值。扇区的大小为定义在 **<sys/param.h>** 之中的 **DEV_BSIZE** 。

名称	类型	说明
ns	***num***	每个磁道的扇区数
nt	***num***	每个柱面的磁道数
nc	***num***	磁盘的总柱面数
b0	***num***	块大小（字节）
f0	***num***	片断大小（字节）
s0	***num***	用扇区数表示的磁盘大小
rm	***num***	每分钟旋转

举例：

```
HP_7914:
    :132.1 MB:ns#16:nt#7:nc#1152:\
    :s0#129024:b0#8192:f0#1024:\
    :se#256:rm#3600:
```

作者

disktab 由 HP 和加州大学伯克利分校联合开发。

文件

/etc/disktab

另请参阅

newfs(1M)、 getdiskbyname(3C)。

名称

dlpi.h - 数据链路提供程序接口标准头文件

概要

/usr/include/sys/dlpi.h

说明

<dlpi.h> 是标准的头文件，它包含 DLPI 2.0 标准规定的 DLPI 请求。它包含对基元、应答和相关结构的定义。

所有想要通过 DLPI 与 LAN 驱动程序进行交互的 DLS 用户（用户空间和内核空间）必须包含这个头文件。

头文件包含对连接模式和无连接服务的定义。

头文件 **<dlpi_drv.h>** 提供网络设备驱动程序与 DLPI 进行交互所要求的定义，请参阅 *dlpi_drv*(4) 。头文件 **<dlpi_ext.h>** 提供 HP 专用的 DLPI 2.0 标准扩展；请参阅 *dlpi_ext*(4) 。联机帮助页 *dlpi*(7) 提供了 HP-UX 的 DLPI 概要。

作者

<dlpi.h> 由 HP 创建，基于 DLPI 2.0 标准。

另请参阅

dlpi_drv(4)、 dlpi_ext(4)、 dlpi(7)、 lan(7)。

《DLPI Programmer's Guide》，2003，Hewlett-Packard
《Driver Development Guide》，Hewlett-Packard

名称

dlpi_drv.h - 设备驱动程序与 DLPI 交互接口定义

概要

/usr/include/sio/dlpi_drv.h

说明

<dlpi_drv.h> 头文件包含一些结构和函数原型的定义；网络设备驱动程序需要使用这些结构和函数原型与 DLPI（数据链路提供程序接口）进行交互。

头文件包含紧密耦合和松散耦合驱动程序所使用的接口。DLPI 用作紧密耦合驱动程序与 DLS 用户之间的单独接口。但是，松散耦合驱动程序仅仅依赖于 DLPI 向用户空间命令 *lanscan*(1M) 提供信息以进行显示。

所有松散耦合和紧密耦合驱动程序都必须包含此文件。

联机帮助页 *dlpi*(7) 提供 HP-UX 的 DLPI 概要。头文件 **<dlpi.h>** 包含 DLPI 请求，请参阅 *dlpi*(4) 。头文件 **<dlpi_ext.h>** 提供 HP 特有的 DLPI 2.0 标准扩展，请参阅 *dlpi_ext*(4) 。

警告

在以后版本的 HP-UX 中， **<dlpi_drv.h>** 中提供的接口可能会有所变化。

作者

<dlpi_drv.h> 由 HP 开发。

另请参阅

lanscan(1M)、 dlpi(4)、 dlpi_ext(4)、 dlpi(7)、 lan(7)。

《Driver Development Guide》， Hewlett-Packard
《Device Driver Reference》， Hewlett-Packard
《DLPI Programmer’s Guide》， 2003， Hewlett-Packard

名称

dlpi_ext.h - HP 特有的 DLPI 扩展

概要

/usr/include/sys/dlpi_ext.h

说明

<dlpi_ext.h> 是 HP 特有的数据链路提供程序接口（DLPI 2.0 标准）扩展的头文件。头文件包含对基元、应答、ioctls 和相关结构的定义，以满足 DLS 用户需要但 DLPI 2.0 标准不能提供的要求。

所有想要通过 DLPI 与 LAN 驱动程序进行交互的 DLS 用户（用户空间和内核空间）都必须包含这个头文件。

联机帮助页 *dlpi*(7) 提供了 HP-UX 的 DLPI 概要。头文件 **<dlpi.h>** 包含 DLPI 请求；请参阅 *dlpi*(4) 。**<dlpi_drv.h>** 头文件包含网络设备驱动程序与 DLPI 交互所需的定义；请参阅 *dlpi_drv*(4) 。

作者

<dlpi_ext.h> 由 HP 开发。

另请参阅

dlpi(4)、 dlpi_drv(4)、 dlpi(7)、 lan(7)。

《DLPI Programmer’s Guide》，2003，Hewlett-Packard
《Driver Development Guide》，Hewlett-Packard

名称

DOSIF - DOS 交换格式描述

说明

命令 **doschmod** 、 **doscp** 、 **dosdf** 、 **dosls** 、 **dosll** 、 **dosmkdir** 、 **dosrm** 和 **dosrmdir** 将从 HP-UX 中去除。使用 **dos2ux** 和 **ux2dos** 命令来在 HP-UX 和 DOS 文件格式之间进行文件转换：请参阅 *dos2ux*(1) 。

DOSIF（DOS 交换格式）是 DOS 操作系统使用的媒体格式的名称。该格式基于 IBM PC 和 PC AT 以及 HP Vectra 系统的使用。

在第一部分中描述的 DOS 实用程序（此后称为 *dos**(1) ）是为了从 DOSIF 卷中读取数据以及向 DOSIF 卷中写入数据而提供。使用这些工具来从 DOSIF 卷中检索信息。

*dos**(1) 实用程序是唯一可以直接与 DOSIF 卷的内容进行交互的 HP-UX 命令。与 DOSIF 卷的内容进行交互的其他办法是使用 HP-UX DOS 模拟，或者使用象 SoftPC 或 DOS 协处理器那样的协处理器。不能在 DOSIF 卷上使用 **mount** 因为操作系统不能识别它（请参阅 *mount*(1M) ）。

为 *dos**(1) 命令构建文件名时，以 DOSIF 卷的 HP-UX 路径名开头，然后添加一个冒号 (**:**) ，之后是文件名：

*device_file***:***file*

或者

*path_name***:***file*

注释：此文件命名惯例只适用于 *dos**(1) 实用程序的参数。它对 HP-UX 应用程序中的任何其他使用不构成合法路径名。

在指定 HP-UX 和 DOS 文件名时，都可以使用元字符 ***** 、 **?** 和 **[...]** 。由于必须通过 DOS 使用程序，而不是Shell来执行文件名扩展，因此在指定 DOS 文件名时，必须将这些符号用引号引起来。 *dos**(1) 实用程序扩展文件名的方式如模式匹配表示法 *regexp*(5) 中所述。

根据惯例，如果指定了 HP-UX 设备名和结尾的冒号，但是没有提供文件或目录名（例如， **/dev/rdsk/c1t1d0:**），则假定为 DOS 文件系统的根 (*/*) 。

举例

指定 DOSIF 文件 **/dos/ivy** ，该文件通过 HP-UX 专用文件 **/dev/rdsk/c1t1d0** 访问：

/dev/rdsk/c1t1d0:/dos/ivy

指定 DOSIF 文件 **/math** ，该文件通过保存为 HP-UX 文件 **/home/mydir/driveC** 的 DOS 卷访问：

/home/mydir/driveC:/math

另请参阅

dos2ux(1)、 doschmod(1)、 doscp(1)、 dosdf(1)、 dosls(1)、 dosmkdir(1)、 dosrm(1)。

名称

dp - 供 DDFA 软件和 Telnet 端口使用的专用端口文件标识功能

说明

dp 文件有两种用途：

数据通信和终端控制器设备文件访问

dp 文件供数据通信和终端控制器设备文件访问 (DDFA) 软件使用，使得 HP-UX 应用程序按照与直接连接到 HP-UX 系统的设备相同的方式，按程序指令访问终端服务器端口。对于每一个已配置的终端服务器端口，该文件有一个（对应的）单行条目。

dp 文件包含 DDFA 软件用于设置和管理到指定终端服务器端口的出站连接所需的信息。专用端口分析器 (**dpp**) 分析该文件，为文件中指定的每一个出站连接衍生出站连接守护程序 (**ocd**) 。

Telnet 端口标识

dp 文件供 HP-UX Telnet 守护程序 (**telnetd**) 使用，用于标识来自 HP 数据通信和终端控制器 (DTC) 的 telnet 连接的调用端口和电路板。

连接时，主机协商 telnet 环境选项，而 DTC 应答其连接设备的端口和电路板编号。 **Telnetd** 将端口和电路板编号映射到已提前在 **dp** 文件中配置的设备已知的名称。

数据通信和终端控制器设备文件访问

对于出站连接，条目应具有下述格式：

*dtc_name board***/***port pseudonym config_file log_level*

下面给出了每一个字段的具体内容。

Telnet 端口标识

要配置 **dp** 文件以便使用 Telnet 端口标识功能，应将缺省文件 **/usr/examples/ddfa/dp** 复制到一个新文件，且该副本已为入站连接配置了适当值。建议的步骤是创建一个目录，用以存放 **dp** 文件和已修改的端口配置文件。

此用途的条目应具有下述格式：

*dtc_name board***/***port pseudonym*

下面给出了每一个字段的具体内容。

配置信息

有三种指定终端服务器端口的方法：

- 显式指定端口的 IP 地址。
- 指定节点名或 DTC 的 IP 地址，然后指定电路板和端口。
- 指定节点名或终端服务器的 IP 地址，以及该端口的 TCP 端口服务地址。

可在 **dp** 文件中追加注释，方法是以 **#** 字符作为注释的开头。分析器将忽略 **#** 字符后面的任何内容。 **dp** 文件中的字段由空格字符分隔。

有关如何配置 DDFA 软件的详细信息，请参阅 *ddfa*(7) 。

dp 文件中条目的字段如下所示：

dtc_name 该字段是节点名，或是正被访问的终端服务器的 IP 地址，或是终端服务器中端口的 IP 地址。节点名必须在名称数据库中定义。

*board***/***port* 该字段包含终端服务器端口地址，地址中的各部分由 **/** 字符分隔。没有必要在值的前面填充零。该端口地址由 **ocd** 检查，而非 **dpp** 。 *board* 的有效值是 0 至 7， *port* 的有效值是 0 至 31（相反，若指定 TCP 端口服务地址，则不应用这些限制）。

如果 *dtc_name* 字段显式定义节点名或终端服务器端口的 IP 地址，则 *board***/***port* 字段中的值必须是 **xx/xx** （使用 **X** 或 **x** ）。

如果该字段的形式为 **xx/***n* ，其中 *n* 是一个十进制数字，则假定 *n* 是 TCP 端口服务地址，建立连接时使用该地址。

pseudonym 此字段是系统和终端用户应用程序可识别的设备文件绝对路径。路径名中的设备文件名部分仅限于 14 个字符。

pc_file_path 该字段是一个端口配置文件路径，其中包含了终端服务器端口的配置信息。此字段是出站连接所必需的，因为 **dpp** 将此字段用作其标志，以便为该条目衍生一个守护程序。

log_level 此字段是特定 **ocd** 的日志级别，它决定了发送给 **/var/adm/syslog** 的消息严重性级别。日志级别（及如何将它们与系统日志级别关联起来）如下所示：

0 仅记录 LOG_CRIT 消息。
1 仅记录 LOG_CRIT 和 LOG_ERR 消息。
2 仅记录 LOG_CRIT、LOG_ERR 和 LOG_WARNING 消息。
3 记录所有消息。

它是可选的，可仅为出站连接指定此字段。如果省略此字段，则日志级别设置为 1。

举例

下述各示例说明了文件条目语法。

打印机以 IP 地址 11.234.87.123 被连接到一个 DTC 中 3 号电路板的 1 号端口上。使用设备文件 **/dev/telnet/lp1_ocd** ， HP-UX 后台打印程序可访问附加到该端口的设备。

11.234.87.123 03/01 /dev/telnet/lp1_ocd /usr/examples/ddfa/pcf

打印机以 IP 地址 11.234.87.124 被连接到一个终端服务器端口上。 *board***/***port* 字段包含 **xx/xx** 。使用设备文件 **/dev/telnet/lp2_ocd** ， HP-UX 后台打印程序可访问附加到该端口的设备。

11.234.87.124 xx/xx /dev/telnet/lp2_ocd /usr/examples/ddfa/pcf

打印机被连接到一个使用终端服务器的 TCP 端口服务地址 5001 所访问的端口，且打印机的 IP 地址为 11.234.87.215。使用设备文件 **/dev/telnet/lp3_ocd** ， HP-UX 后台打印程序可访问附加到该端口的设备。

11.234.87.215 xx/5001 /dev/telnet/lp3_ocd /usr/examples/ddfa/pcf

终端被连接到一个 DTC 中 2 号电路板的 1 号端口上，该终端的 IP 地址为 11.234.87.215，希望使用 Telnet 端口标识。

11.234.87.215 02/01 /dev/telnet/tm02

警告

要确保命令（如 *ps* ）显示正确的设备文件名（也就是 *pseudonym* ），所有的别名应该置于 **/dev/telnet** 目录中。如果未在此目录中指定别名的放置位置，则不保证许多命令能够正确显示设备文件名。

此外，要确保命令（如 **w** 、 **passwd** 、 **finger** 和 **wall** ）正确工作，在命令的前 17 个字符中每一个别名必须是唯一的（包括目录前缀 **/dev/telnet/** ）。如果在前 17 个字符中别名不是唯一的，则不保证许多命令能够正确执行其功能。

文件

/usr/sbin/dpp

/usr/sbin/ocd

/usr/sbin/ocdebug

/var/adm/dpp_login.bin

/var/adm/utmp.dfa

/usr/examples/ddfa/dp

/usr/examples/ddfa/pcf

另请参阅

dpp(1M)、ocd(1M)、ocdebug(1M)、syslog(3C)、pcf(4)、ddfa(7)。

名称

efi - 可扩展的固件接口描述

说明

EFI （可扩展的固件接口）是 HP-UX 与基于 Itanium 平台的固件间的接口。可扩展的固件接口支持的文件系统基于 FAT 文件系统。 EFI 使用了系统分区所使用的 FAT-32 和可删除介质所使用的 FAT-12 和 FAT-16。系统分区需要基于 Itanium 平台的可引导磁盘。

对于硬盘，系统分区是一个磁盘扇区的连续分组，其中起始扇区和分区大小由 EFI 分区表（位于硬盘的第二逻辑块）和（或）主引导记录（MBR，位于硬盘的首个扇区）指定。对于软盘，分区被定义为整个磁盘。

系统分区可以包含目录、数据文件和 EFI 映像。 EFI 系统固件可以搜索 EFI 系统分区的 **\EFI** 目录和 EFI 卷，以寻找可加载的 EFI 映像。 HP-UX 的引导装入器是一个典型的 EFI 映像。

HP-UX 包含一组 EFI 实用程序：

efi_fsinit(1M)	初始化 EFI 卷；创建一个标题和一个空目录。
efi_cp(1M)	将文件复制到 EFI 卷和从 EFI 卷中复制出文件。
efi_mkdir(1M)	在 EFI 卷中创建目录。
efi_ls(1M)	列出 EFI 卷的内容。
efi_rm(1M)	从 EFI 卷中删除文件。
efi_rmdir(1M)	从 EFI 卷中删除目录。

在 EFI 卷的内部结构已知的 HP-UX 中， EFI 实用程序是唯一的实用程序。对于其他 HP-UX， EFI 系统分区仅仅是包含未指定数据的简单分区。目前 HP-UX 不能挂接 EFI 卷。

可以使用支持随机访问（通过 *lseek*(2) ） HP-UX 文件（常规磁盘文件或设备专用文件）创建 EFI 卷。在 EFI 卷内，文件和目录用包含 1-255-个字符的文件名标识。文件名可以由任何字母数字字符（A-Z、a-z 或 0-9）组成，文件名还可以包含特殊字符（ “.” 、 “$” 、 “%” 、 “’” 、 “-” 、 “_” 、 “@” 、 “~” 、 “‘” 、 “!” 、( ) + , : ; = # & “?” 、 “^” 、 “[” 、 “]” 、 “{” 、 “}” 和空格） 。EFI 文件名的首个字符可以是除空格外的任何有效 EFI 字符。比较两个 EFI 文件名时，并不区分字母数字字符的大小写。例如，以下文件名是相同的：

ABC_file
abc_file

如果存在其中一个文件名，用户将不能创建另外一个文件名。

目录可以由多个部分组成，中间用斜线 (**/**) 分隔。目录的最后一部分必须后接斜线，以分隔目录和文件名。另有两类特殊目录，(**.**) 和 (**..**)。它们表示当前目录和和父目录，这与其他文件系统相同。

另请参阅

efi_cp(1M)、 efi_fsinit(1M)、 efi_ls(1M)、 efi_mkdir(1M)、 efi_rm(1M)、 efi_rmdir(1M)。

名称

exports、xtab - 需要导出到 NFS 客户端的目录

概要

/etc/exports

/etc/xtab

说明

文件 **/etc/exports** 描述了可以导出给 NFS 客户端的目录。系统管理员使用文本编辑器创建此文件。 **mountd** 每当接收到一个 mount 请求时就会对其进行处理（请参阅 *mountd*(1M)）。

/etc/exports 会被 **exportfs** 命令自动读取（请参阅 *exportfs*(1M)）。如果修改了此文件，则必须运行 **exportfs** (**exportfs -a**) 才能改变守护程序的操作。

如果在启动时此文件已存在，则 **/sbin/init.d/nfs.server** 脚本将执行 **exportfs** 命令并导出文件中列出的文件系统。

/etc/xtab 包含了当前导出目录中的条目。此文件只能由程序使用 **getexportent** 来访问（使用 **exportfs -u** 将条目从文件中删除）。

目录的一个条目包含如下形式的命令行:

directory **-** *option*[**,** *option*] ...

这里 *directory* 是一个目录（或文件）的路径名称。

options 可以是下列值或者形式之一:

ro 以只读方式导出目录。如果没有指定，目录将以读写方式导出。 **ro** 和 **rw** 选项不能在同一个导出行中使用。

rw=*hostname*[**:***hostname*]...
以 read-mostly 形式导出目录。 Read-mostly 表示对大多数机器是只读的，但是对指定的机器是读写的。如果没有指定 **ro** 和 **rw** ，这个目录会对所有的机器以读写方式导出。 **ro** 和 **rw** 选项不能在同一导出行中使用。最多可以指定 256 个 *hostnames* 。当在 nsswitch“主机”条目中配置了用于 DNS 命名的服务器时，任何主机都必须表示为一个完全限定的 DNS 名称。当前的 HP-UX 会尝试匹配一个非完全限定的主机名称；这个 HP 特有的功能将在 HP-UX 以后的版本中过时。

anon=*uid* 如果一个请求来自一个未知的用户，则可使用 *uid* 作为有效用户 ID。 注： NFS 服务器总将超级用户 (uid 0) 看作是“未知的”，除非超级用户包含在 **root** 选项中。

此选项的缺省值是 **-2** 。将 **anon** 设置为 **-1** 可禁用匿名访问。

root=*hostname*[**:***hostname*]...
仅对指定 *hostname* 的超级用户赋予根目录访问权限。缺省情况下任何主机都没有对根目录的访问权限。至多可以指定 256 个 *hostname* 。此列表上的各个 *hostname* 不能保证可成功挂接指定的文件系统。如果指定了一个非空的访问列表，则 *hostname* 还必须满足下述 *access_list*

标准之一。至多可以指定 256 个 *hostnames* 。当在 nsswitch“主机”条目中配置了用于 DNS 命名的服务器时，任何主机都必须表示为一个完全限定的 DNS 名称。当前的 HP-UX 会尝试匹配一个非完全限定的主机名称；这个 HP 特有的功能将在 HP-UX 以后的版本中过时。

access=[*access_list*][**:***access_list*]...

为列出的每一个 *access_list* 条目赋予挂接权限。请参阅下面的“访问列表”小节。一个空的 **access=** 列表允许所有的机器挂接指定的挂接点。

access_list

access_list 参数是一个由冒号分隔的列表，它的组件可以是下列元素中的任意多个：

hostname

一个主机的名称。当在 nsswitch“主机”条目中配置了用于 DNS 命名的服务器时，任何主机都必须表示为一个完全限定的 DNS 名称。当前的 HP-UX 会尝试匹配一个非完全限定的主机名称；这个 HP 特有的功能将在 HP-UX 以后的版本中过时。

netgroup

一个网络组包含多个主机名称。当在 nsswitch“主机”条目中配置了用于 DNS 命名的服务器时，任何主机都必须表示为一个完全限定的 DNS 名称。

DNS suffix

要使用域成员，服务器必须使用 DNS 将主机名称解析为 IP 地址；也就是说，**/etc/nsswitch.conf** 中的“主机”条目必须在指定“nis”或者“nisplus”之前指定“dns”，因为只有 DNS 返回了主机的完整域名。其他名称服务例如 NIS 或者 NIS+ 不能用于在服务器上解析主机名称，因为在将 IP 映射到主机名称时，它们不会返回域信息。例如，

NIS 或者 NIS+

129.144.45.9 --> “myhost”

DNS

129.144.45.9 --> “myhost.myd.myc.com”

使用一个前缀点来区分 DNS 后缀与主机名和网络组名。一个点本身将匹配“myhost”而不匹配“myhost.myd.mycy.com”。此单点功能可用于匹配由 NIS 和 NIS+ 解析的主机，而不能匹配由 DNS 解析的主机。

network

网络或子网组件前面有个 at 符号 (@)。它可以是一个名称，也可以是一个点分地址。如果是一个名称，则可由 **getnetbyname** 将其转换为点分地址（请参阅 *getnetent*(3N)）。**/etc/networks** 中的条目必须顺序包含所有四个八位字节才可有效。

网络前缀假设一个八位字节对齐的网络掩码可以从网络地址中低位部分为零的八位字节中确定。在网络前缀不是字节对齐的情况下，语法允许显式的在斜线 (/) 分界符后指定一个掩码长度。这里掩码是对应 IP 地址中从左起连续有效位的个数。

- 一个前缀的减号 (-) 拒绝对访问列表中相应组件的访问。此列表会被顺序查找，直至匹配到允许或者拒绝访问的项目，或到达列表结束位置时为止。这个选项只有在和主机名、网络和 DNS 前缀一起使用时才有效。如果在主机名前加了一个前缀，并配置为 DNS 命名，则必须完全限定主机名。

async 指定 **async** 可增强 NFS 服务器上的写入性能，因为这样可在 NFS 服务器上实现异步写入。**async** 选项可以在命令行 *directory* 后面的任何地方指定。在使用此选项之前，请参阅下面的警告。

文件中任何地方出现的 **#** 字符，表示从此开始直至行末尾都是注释。

没有附带名称列表的目录名称允许任何机器挂接指定目录。

/etc/exports 包含了文件系统列表和允许远程挂接每个文件系统的 **access_lists** 或者机器名称。文件系统名称是左对齐的，并且后面跟有由空格分隔的名称列表。一个没有附带名称列表的文件系统名称，意味着文件系统对每个人都是可用的。

文件中任何地方出现的 **#**，表示从此开始至此行末尾都是注释。

举例

```
/usr/games cocoa fudge              # export to only these machines
/usr     -access=clients            # export to my clients
/usr/local                          # export to the world
/usr2    -access=bison:deer:pup     # export to only these machines
/var/adm  -root=bison:deer          # give root access only to these
/usr/new  -anon=0                   # give all machines root access
/usr/temp -rw=ram:alligator         # export read-write only to these
/usr/bin  -ro                       # export read-only to everyone
/usr/stuff -access=bear,anon=-2,ro  # several options on one line
/usr/subnet -access=@mysubnet       #use mysubent in /etc/networks
/usr/subnet1 -access=@192.5         #clients must be in the 192.5.0.0 subnet
/usr/domain -access=.myd.myc.com    #clients must be in .myd.myc.com
/usr/restrict -access=-host1.myd.myc.com:sales    # disallow -host1 in the sales netgroup.
```

警告

如果使用了 **async** 选项，则 只有在进行写操作，并且 只有 NFS 服务器在将写应答发送给客户端之后出现故障时，才会出现未报告的数据丢失。特别是，已经在服务器磁盘队列中，但还没有写入磁盘的数据块， 可能会丢失。

您不能导出一个父目录，或者是驻留在 同一文件系统中的已导出目录的子目录。例如，如果 **/usr** 和 **/usr/local** 目

录位于同一个磁盘分区，则不允许同时将它们导出。

作者

exports 由 Sun Microsystems, Inc. 开发。

文件

/etc/exports	静态导出信息
/etc/xtab	当前导出目录状态
/etc/hosts	主机名列表
/etc/netgroup	网络组列表
/etc/networks	网络信息
/sbin/init.d/nfs.server	执行 **exportfs** 命令的脚本。

另请参阅

exportfs(1M)、 mountd(1M)、 nfsd(1M)、 hosts(4)、 netgroup(4)、 networks(4)。

名称

fspec - 文本文件中的格式规范

说明

有时方便地采用非标准制表符维护 HP-UX 系统中的文本文件，（意味着不在每 8 列设置制表符）。通常，这样的文件必须转换为标准格式 –，此格式经常以合适数量的空格替换所有的制表符 –，这样的替换发生在 HP-UX 命令能够处理它们前。文本文件首行的格式规范指定制表符如何扩展到文件的其余部分。

格式规范包括一个参数序列，各参数由空格分隔、由括号 **<:** 和 **:>** 括起。每个参数包含一个键字母，可能紧跟一个值。该命令可以识别下列结构：

t*tabs* **t** 参数为文件指定制表符设置。 *tabs* 类型的值必须是下列之一：

1. 以逗号分隔的列数列表，表示特定列的制表符设置；
2. **-** 紧跟一个整数 *n* ，表示制表符间隔 *n* 列；
3. **-** 后跟一个“固定”制表符规范的名称。

标准制表符由 **t-8** 或相当的 **t1** 、 **9** 、 **17** 和 **25** 等指定。可识别固定制表符由 **tabs** 命令定义（请参阅 *tabs*(1) ）。

s*size* 参数 **s** 指定行最大的大小。 *size* 的值必须是一个整数。制表符扩展后执行大小检查，但在这之前间距被插入至行首。

m*margin* 参数 **m** 指定插入每行行首的空格数。 *margin* 的值必须是一个整数。

d 参数 **d** 没有值。它的存在表示包含格式规范的行将从已转换的文件中删除。

e 参数 **e** 没有值。它的存在表示当前的格式是合适的，仅在文件中遇到另一种格式规范时此情况才发生变化。

缺省值（假设不提供参数）是 **t-8** 和 **m0** 。如果不指定 **s** 参数，则不执行大小检查。如果文件首行不包含格式规范，则假设上述缺省值在整个文件范围内采用。下面是一个包含格式规范的行的示例：

*** <:t5,10,15 s72:> ***

如果格式规范可以看成注释，则不必对参数 **d** 编码。

几个 HP-UX 系统命令正确解释文件的格式规范。它们中 **ed** 可以用于将文件转换为其他 HP-UX 系统命令能够接受的标准格式。

另请参阅

ed(1)、newform(1)、tabs(1)。

名称

fstab - 关于文件系统的静态信息

概要

#include <fstab.h>

说明

fstab 是一个 ASCII 文件，该文件位于 **/etc** 目录中。程序读取该文件，但是不对其进行写操作。系统管理员负责恰当地创建和维护该文件。

/etc/fstab 包含可挂接文件系统条目的一个列表。每个文件系统条目出现在单独一行，并且由以一个或多个空格或制表符分隔的字段组成。

/etc/fstab 中条目的顺序只对没有 *pass number* 字段的条目才是重要的。没有 *pass number* 的条目在有 *pass number* 的条目被检查后由 **fsck** 检查（请参阅 *fsck*(1M)）。

每个文件系统条目都必须包含一个 *device special file* ，此外可能按照如下顺序包含以下所有字段：

directory

type

options

backup frequency

pass number （类似于 **fsck** ）

comment

如果设备专用文件名之后有任何字段，则所有字段必须按照指定的顺序出现，从而确保正确地占位。

来自该文件的条目可以使用 **getmntent()** 访问（请参阅 *getmntent*(3X)）。

这些字段由空白字符分隔，并以 **#** 作为表示注释的条目或字段中的第一个非空白字符。

device special file 块设备专用文件名。该字段被 **fsck** 、 **mount** 、 **swapon** 、 **crashconf** 和其他命令使用以确定文件系统所处的保存设备的位置。

directory 已挂接文件系统的根的名称，该文件系统对应于 *device special file* 。如果 *type* 是 **swapfs** ，则 *directory* 可以是文件系统中任意目录的名称。每个文件系统只需要指定一个目录。*directory* 必须存在并且给出绝对路径名。

type 可以是 **swap** 、 **swapfs** 、 **dump** 、 **ignore** 或文件系统类型（例如， **hfs** 、 **vxfs** 、 **cdfs** 、 **nfs** 或 **lofs** ）。

如果 *type* 是 **swap** ，则 *device special file* 作为交换空间由 **swapon** 命令提供（请参阅 *swapon* (1M)）。 *options* 字段有效。 **swap** 条目忽略 *directory* 、 *pass number* 和 *backup frequency* 。

如果 *type* 是 **swapfs** ，则 *directory* 所处的文件系统成为 **swapon** 可用的交换空间。 *options* 字段有效。 **swapfs** 条目忽略 *device special file* 、 *pass number* 和 *backup frequency* 。

如果 *type* 是 **dump** ，则 *device special file* 作为系统崩溃转储可能发生的区域，由 **crash-conf** 命令设置（请参阅 *crashconf*(1M) ）。 **dump** 条目忽略 *options* 、 *directory* 、 *pass number* 和 *backup frequency* 。

由 *type* **ignore** 标记的条目被所有命令忽略，可用于标记未使用的部分。如果 *type* 被指定为 **ignore** 、 **dump** 、 **swap** 或 **swapfs** ，则该条目被 **mount** 和 **fsck** 命令忽略（请参阅 *mount*(1M) 和 *fsck*(1M) ）。 *fsck* 还忽略 *type* 被指定为 **cdfs** 、 **nfs** 或 **lofs** 的条目。

options 逗号分隔的选项关键字列表，如 *mount*(1M) 或 *swapon*(1M) 中给出的。使用的关键字取决于 *type* 中指定的参数。

backup frequency 为今后备份实用程序可能使用保留。

pass number **fsck** 命令用以决定文件系统检查的顺序。根文件系统应该指定 1 为 *pass number* ，最先被检查，其他文件系统应具有更大的编号。（ **fsck** 命令忽略 *pass number* 为零的文件系统）。

一个驱动器中的文件系统应该分配不同的关口编号，但是不同驱动器上的文件可以在同一关口上进行检查，从而利用硬件中可能的并行处理。如果 *pass number* 不存在，则 **fsck** 在检查了所有符合关口编号条件的文件系统之后，再检查每个这样的文件系统。

comment 可选的字段以 **#** 字符开头并以换行字符结尾。保留从 *pass number* 到 *comment* 字段（如果存在）或者到换行符之间的空间以供今后使用。

/etc/fstab 中的 *device special file* 字段数目没有限制。

网络功能

NFS

如果 *type* 字段是 **nfs** ，则表示远程 NFS 文件系统。对于 NFS 文件系统， *device special file* 应该是“:”之后的服务计算机名称后面是被服务目录的服务计算机路径。 NFS 条目忽略 *pass number* 和 *backup frequency* 字段。

举例

典型 **/etc/fstab** 条目举例:

使用缺省的挂接选项在 **/home** 添加一个 HFS 文件系统（备份频率 0） fsck 关口 2:

/dev/dsk/c0t6d0 /home hfs defaults 0 2 # /home disk

使用 LVM 添加一个 **swap** 设备到被管系统，使用缺省选项（注意， *directory* 字段 (**/**) 不能为空，即使它被忽略）：

/dev/vg01/lv10 / swap defaults 0 0 # swap device

在实现全盘布局的系统上添加交换设备以使用该文件系统末端后的空间 (*options*=**end**):

/dev/dsk/c0t5d0 / swap end 0 0 # swap at end of device

在包含 **/swap** 目录的文件系统上添加文件系统交换空间。 *type* 是 **swapfs** ； 设置 *options* 为 **min=10** 、 **lim=4500** 、 **res=100** 和 **pri=0** （请参阅 *swapon* (1M)）解释 *options* ）。 *device* 字段被忽略但是必须不为空：

default /swap swapfs min=10,lim=4500,res=100,pri=0 0 0

（注意，文件系统条目和交换条目对提供这两种服务的设备来说都是必需的）。

如果系统崩溃则使用设备提供转储空间。 *directory* 字段被忽略但是必须不为空。

/dev/dsk/c0t5d0 / dump defaults 0 0

（注意，交换条目和转储条目对提供这两种服务的设备来说都是必需的）。

相关内容

NFS

这有一个为支持 NFS 文件系统的系统挂接 NFS 文件系统的例子：

server:/mnt /mnt nfs rw,hard 0 0 #mount from server.

作者

fstab 由 AT&T、加州大学伯克利分校和 HP 开发。

文件

/etc/fstab

/usr/include/fstab.h

另请参阅

fsck(1M)、 mount(1M)、 swapon(1M)、 crashconf(1M)、 getfsent(3X)、 getmntent(3X)、 mnttab(4)。

名称

fs_vxfs - VxFS 文件系统卷格式

概要

#include <sys/types.h>

#include <sys/param.h>

#include <sys/fs/vx_fs.h>

说明

VxFS 超级块始终从文件系统开始处字节偏移量为 8192 的地方开始。超级块的位置固定，所以不同系统实用程序均可得知从何处定位。

超级块字段包括下列基本大小和偏移量:

fs_bsize 文件系统的块大小。 VxFS 支持的块大小包括 1024、2048、4096 和 8192 字节。块大小的缺省值取决于创建文件系统时的大小。有关指定值，请参阅 *mkfs_vxfs*(1M) 中的 **bsize** 选项。

fs_ctime 文件系统的创建日期。 **time()** 系统调用提供时间。

fs_defiextsize 块中的间接数据范围的缺省大小。当前缺省设置为 64。

fs_dsize 文件系统中数据块的数目。数据块为可能分配到文件系统中文件的块。

fs_ectime 为了更高的精确度而扩展文件系统的创建日期时出现的占位符。当前为零。

fs_immedlen i 节点中直接数据的大小，以字节为单位。当前为 96。

fs_logend 上一日志区域块的块地址。可以使用 **mkfs** 命令指定日志区域大小。如果未指定，对小于 8 MB 的文件系统，缺省大小为 256 块，大小在 8 MB 至 512 MB 之间的文件系统为 1024 块，大小在 512 MB 以上的文件系统为 16384 块。对于较小的文件系统，为避免浪费空间将减少缺省值。

fs_logstart 第一个日志区域块的块地址。当前为 2。

fs_magic 文件系统的 (**VX_MAGIC**) 幻数。此数字标识文件系统类型为 VxFS 的文件系统。

fs_nau 文件系统分配单元数。

fs_ndaddr **VX_EXT4** 映射类型支持的直接盘区号（请参阅 i 节点列表部分的说明）。当前为 10。

fs_size 文件系统中块的数，大小为 *fs_bsize* 。 **fs_size** 为无符号的 32 位数字，因而 VxFS 文件系统块的最大数限制为 32 位。

fs_version 文件系统磁盘布局的版本号 (**VX_VERSION**) 。

前置字段，用于定义大小和组成文件系统。要减少实用程序的计算需求，部分值由基本值派生，并放置在超级块中。

超级块包含下列派生偏移量:

fs_aublocks 分配单元的数据块数。

fs_aufirst 第一分配单元的地址，以块为单位。在目的日志和第一分配单元之间有间隔。此间隔用于在所需边界对齐第一分配单元。

fs_auemlen 可用盘区映射的长度，以块为单位。

fs_aulen 分配单元的长度，以块为单位。

fs_aupad 分配单元对齐填充的长度，以块为单位。

fs_bmask 用于将偏移量向上舍入到最接近的较小块边界的掩码值 (*byte_offset* **&** *fs_bmask*) 。

fs_boffmask 用于从最接近较小块边界的开始处生成偏移量的掩码值 (*byte_offset* **&** *fs_boffmask*) 。

fs_bshift *fs_bsize* 基数 2 的对数。用于转换字节偏移量为块偏移量。

fs_bstart 从分配单元首端开始的第一数据块偏移量，以块为单位。分配单元标题可能包括用于将第一数据块对齐到指定边界的填充。

fs_checksum 上述字段的校验和。 **VX_FSCHECKSUM** 宏验证或计算校验和。

fs_emap 从分配单元首端开始的可用盘区映射 (emap) 偏移量，以块为单位。

fs_fbstart 从文件系统首端开始的第一数据块偏移量，以块为单位。

fs_femap 从文件系统首端开始的第一可用盘区映射 (emap) 偏移量，以块为单位。

fs_fimap 从文件系统首端开始的第一个可用 i 节点映射 (imap) 偏移量，以块为单位。

fs_iaddrlen 间接地址块的大小，以块为单位。间接地址块大小为 8K 字节。此字段设置为 (8K / *fs_bsize*) 。

fs_imap 从分配单元首端开始的可用 i 节点映射 (imap) 偏移量，以块为单位。

fs_inopau 分配单元中的 i 节点数。

fs_inopb 在 i 节点列表中，每个 **fs_bsize** 块的 i 节点条目数。 VxFS i 节点当前长度为 256 字节。

fs_inoshift *fs_inopb* 基数 2 的对数。用于将 i 节点列表中的 i 节点数转换为块偏移量。

fs_maxtier **fs_aublocks** 基数 2 的对数。

fs_nindir 间接地址盘区中的条目数。间接地址盘区当前长度为 8192 字节，则 **fs_nindir** 值为 2048。

创建文件系统时前置字段初始化，并保持不变，除非文件系统大小改变。该字段复制到每个分配单元的标题。

下列为动态添加字段:

fs_clean 文件系统挂接为读/写访问时，设置为 **VX_DIRTY** 。在 **umount** 或成功 **fsck** 时设置为 **VX_CLEAN** 。文件系统无法挂接为读/写访问，除非 **fs_clean** 字段为 **VX_CLEAN** 。

fs_efree 在文件系统中，当前每个盘区大小的可用盘区数的数组。

fs_firstlogid 文件系统挂接时的初始登录 ID。

fs_flags 可识别下列标志：

当文件系统从错误中恢复后需要完整结构检查时，设置 **VX_FULLFSCK** 。如果设置了此标志，在回放恢复结束后执行完整检查。

VX_NOLOG 文件系统与 **VX_MS_NOLOG** 选项挂接时设置。如果设置了此标志，不执行日志回放恢复。

VX_LOGBAD I/O 错误导致日志无效时设置。如果设置了此标志，不执行日志回放恢复。

VX_LOGRESET 登录 ID 运行超过 **VX_MAXLOGID** (2^30) 时设置。在下一合适时间登录 ID 重置（例如挂接或系统同步）。

VX_RESIZE 文件系统进行重设大小时设置。如果 **fsck** 检测到此标志，则执行重设大小恢复。有关文件系统扩展，请参阅 *fsadm_vxfs*(1M) 。

VX_UPGRADING 进行文件系统升级时设置。如果 **fsck** 检测到此标志，则执行升级操作。

fs_fname 文件系统名称（6 个字符）。

fs_fpack 文件系统数据包标签（6 个字符）。

fs_free 可用数据块数。

fs_logversion 日志格式的版本号。由每个挂接的内核设置，用于确认 **fsck** 所运行的日志回放可以理解内核写入的日志格式。

日志格式可以根据不同版本更改，所以全部文件系统在升级到一个新版本之前需要清除。

fs_mod 挂接文件系统修改时设置。在执行同步操作时，指示超级块是否需要重新写入。

fs_reserved 保留以备将来使用。

fs_time 上次超级块写入磁盘时间，以秒和微秒数指示从 1970 年 1 月 1 日 GMT 0:00:00 至今的时间。

在 VxFS 版本 2 磁盘布局及更高版本中需要下列字段。该字段在文件系统创建时设置，并且保持不变。复制到每

个分配单元的标题。

fs_checksum2 字段校验和。

fs_dinosize 磁盘 i 节点的大小，以字节为单位。当前为 256 字节。

fs_dniaddr 每个 i 字节的间接地址等级数。

fs_iauimlen 在 i 节点分配单元中可用 i 节点映射的长度，以块为单位。

fs_iausize i 节点分配单元的大小，以块为单位。

fs_oltsize **fs_oltext** 的对象位置表盘区指向大小，以块为单位。

fs_oltext 两个盘区地址的数组。这些盘区地址指向第一对象位置表盘区的两个副本。

另请参阅

fsadm(1M)、 fsadm_vxfs(1M)、 fsck(1M)、 fsdb(1M)、 mkfs(1M)、 mount(2)、 time(2)、 inode_vxfs(4)。

名称

ftpaccess - ftpd 配置文件

概要

/etc/ftpd/ftpaccess

说明

/etc/ftpd/ftpaccess 文件用于配置 **ftpd** （请参阅 *ftpd*(1M) ）的操作。

访问功能

autogroup *groupname class* [*class* ...]

如果 **anonymous** 用户是任何 *class* 的成员，则 FTP 服务器将针对 *groupname* 执行 **setgid()** 。这允许特定类的匿名用户访问供组和所有者只读的文件和目录。 *groupname* 是 **/etc/group** （或者 **getgrent()** 库例行程序所使用的任何机制；请参阅 *getgrent*(3C) ）中的有效组。

class *class typelist addrglob* [*addrglob* ...]

定义用户的 *class* ，其源地址形式为 *addrglob* 。可以定义 *class* 的多个成员。有多个 **class** 命令可以列出该类的其他成员。如果可以向当前会话应用多个 **class** 命令，则将使用列在访问文件中的第一个命令。如果无法为主机定义有效的类，则将导致访问被拒绝。 *typelist* 是以下任何关键字的逗号分隔列表： **anonymous** 、 **guest** 和 **real** 。如果包括 **real** 关键字，则该类可以与使用 FTP 访问实际帐户的用户相匹配；如果包括 **anonymous** 关键字，则该类可以与使用匿名 FTP 的用户相匹配。 **guest** 关键字与来宾访问帐户相匹配（有关详细信息，请参阅下面的 **guestgroup** ）。

addrglob 可以是已进行匹配替换的域名或数字地址。该指令可以有多个 *addrglob* 。为了避免在有多个 *addrglob* 时发生混淆，可以将所有的 *addrglob* 放在一个文件中，并指定该文件的路径来代替 *addrglob* 的路径。

在 *addrglob* 之前放置一个感叹号 (!) 会抵消测试。例如：

class rmtuser real !*.example.com

会将来自 **example.com** 域外部的实际用户归为 **rmtuser** 类。使用该选项时一定要小心。请记住，每个测试的结果都与该行上的其他测试构成 OR （“或”）的关系。

deny *addrglob message_file*

始终拒绝访问与 *addrglob* 相匹配的主机。 *message_file* 文件中的拒绝消息将显示给被拒绝访问的主机。 *addrglob* 可以是 **!nameserved** ，以便拒绝对没有工作名称服务器的站点的访问。它还可以是以斜线 (**/**) 开头且包含其他地址匹配替换形式的文件名，也采用 *address:netmask* 或 *address/cidr* 形式。

guestgroup *groupname* [*groupname* ...]
guestuser *username* [*username* ...]
realgroup *groupname* [*groupname* ...]
realuser *username* [*username* ...]

对于 **guestgroup** ，如果 **real** 用户是任何 *groupname* 的成员，则会话将按照与匿名 FTP 完全相同的方式进行设置。换言之，在执行 **chroot()** 之后，将不再允许用户发出 **USER** 和 **PASS** 命令。 *groupname* 是 **/etc/group**

（或者 **getgrent()** 库例行程序所使用的任何机制）中的有效组。

必须按照与匿名 FTP 完全相同的方式来正确设置用户的主目录。passwd 条目的主目录字段分成两个目录。第一个字段是将成为 **chroot** 调用的参数的根目录。第二部分是相对于根目录的用户主目录。这两部分用 **/./** 分隔。

例如：

在 **/etc/passwd** 文件中，示例条目为：

guest1:<passwd>:100:92:Guest Account:/ftp/./incoming:/etc/ftponly

当 **guest1** 成功登录时，FTP 服务器将执行 **chroot (/ftp)** ，然后执行 **chdir (/incoming)** 。来宾用户将只能访问 **/ftp** 下的目录结构（对于 **guest1** ，它将充当并显示为 **/** ），如同匿名 FTP 用户一样。

组名可以由名称或数字 ID 来指定。要使用数字组 ID，请在数字前面放置一个 **%** 。可以给出数字的范围。使用星号 (*) 表示所有的组。

guestuser 的工作方式与 **guestgroup** 相似，但是它使用用户名（或数字 ID）。

realuser 和 **realgroup** 具有相同的语法，但是它们的效果与 **guestuser** 和 **guestgroup** 截然相反。它们在远程用户确定为来宾时允许实际用户进行访问。例如：

guestuser *
realgroup admin

会导致将所有非匿名用户视为来宾，但管理组中被授予实际用户访问权限的用户除外。

nice *nice-delta* [*class*]

如果远程用户是命名 *class* 的成员，则按照所指示的 *nice-delta* 值调整 ftpd 服务器进程的 nice 值。如果未指定 *class* ，则将 *nice-delta* 用作 ftpd 服务器进程 nice 值的缺省调整。这个缺省的 nice 值调整仅用于调整某些用户的服务器进程的 nice 值，对于这些用户所属的类来说，ftpaccess 文件中没有类特定的 **nice** 指令。

defumask *umask* [*class*]

如果远程用户是命名类的成员，则设置应用于守护程序所创建文件的 *umask* 。如果未指定 *class* ，则将 *umask* 用作未指定掩码的类的缺省值。

keepalive { **yes|no** }

为数据套接字设置 TCP **SO_KEEPALIVE** 选项。 **keepalive** 可用于控制对网络连接的断开。 **yes** 表示设置 TCP **SO_KEEPALIVE** 选项。如果使用 **no** ，则行为取决于系统的缺省设置（请参阅 *ndd*(1M) ）。

注释：建议将 **keepalive** 设置为 **yes**，以便使网络通信保持连接状态。

timeout accept [*seconds*]

timeout connect [*seconds*]

timeout data [*seconds*]

timeout idle [*seconds*]

timeout maxidle [*seconds*]

timeout RFC931 [*seconds*]

设置各种超时值。

accept [*seconds*] （缺省值为 120 秒）。指定守护程序将等待传入的 (PASV) 数据连接的时间。

connect [*seconds*] （缺省值为 120 秒）。指定守护程序在试图建立传出 (PORT) 数据连接时将等待的时间。这将影响实际的连接尝试。守护程序会进行多次尝试，并在每次尝试之间休眠一会儿，然后完全放弃。

data [*seconds*] （缺省值为 1200 秒）。指定守护程序将等待数据连接上某个活动的时间。建议将此值设置为较高的值，因为远程客户端的链接速度可能较慢，所以该客户端可能会有相当多的数据排在队列中。

idle [*seconds*] （缺省值为 900 秒）。指定守护程序将等待下一个命令的时间。使用 **ftpd** （请参阅 *ftpd*(1M)）的 **-t** 选项，可以覆盖缺省值（900 秒）。如果指定了 **idle**，则该值将覆盖缺省值以及用 **ftpd** 的 **-t** 选项设置的值。SITE IDLE **ftpd** 命令允许远程客户端为空闲超时值设置更高的值。

maxidle [*seconds*] （缺省值为 1200 秒）。指定空闲超时值的最大秒数。使用 **ftpd** （请参阅 *ftpd*(1M)）的 **-T** 选项，可以覆盖缺省值（1200 秒）。如果指定了 **maxidle**，则该值将覆盖缺省值以及用 **ftpd** 的 **-T** 选项设置的值。

RFC931 [*seconds*] （缺省值为 10 秒）。指定守护程序允许执行整个 RFC931 (AUTH/ident) 转换的最长时间。如果将它设置为零 (0) 秒，则将完全禁止守护程序使用此协议。通过 RFC931 获取的信息将记录在系统日志中，该信息实际上不用于任何验证中。

file-limit [*raw*] { **in|out|total** } *count* [*class*]

限制给定 *class* 中的用户可以传输的数据文件的数量 (*count*)。可以针对 **in**、 **out** 或 **total** 文件设置限制。如果未指定 *class*，则该限制是未指定限制的类的缺省值。可选的 *raw* 参数会将限制应用于总流量（而不仅仅是数据文件）。

data-limit [*raw*] { **in|out|total** } *count* [*class*]

限制给定类中的用户可以传输的数据字节数。可以针对 **in**、 **out** 或 **total** 字节设置限制。如果未指定 *class*，则该限制是未指定限制的类的缺省值。可选的 *raw* 参数会将限制应用于总流量（而不仅仅是数据文件）。

limit-time { ***|anonymous|guest** } *minutes*

限制会话可以占用的总时间。缺省情况下没有任何限制。从来不对实际用户进行限制。

guestserver [*hostname*] ...

控制哪些主机可用于匿名访问或来宾访问。如果在没有 *hostname* 的情况下使用，则会拒绝对该站点进行的所有来宾或匿名访问。可以指定多个 *hostname* 。将只允许在命名计算机上进行来宾和匿名访问。如果访问被拒绝，则系统将要求用户使用列出的第一个 *hostname* 。

limit *class n times message_file*

在指定 *times* 将 *class* 限制在 *n* 个用户，并在用户被拒绝访问时显示 *message_file* 。限制检查仅在登录时执行。如果将多个 **limit** 命令应用于当前会话，则将使用第一个适用的命令。无法定义有效的限制或者限制为 **-1** 等同于无限制。 *times* 可以采用下列任一格式：

Any	任何工作日
Fr	星期五
Any0900-1300	一周中的某天，9.00 至 13.00 点。
Th|Any0900-1300	星期四或者 9.00 至 13.00 点。

noretrieve [**absolute|relative**] [**class=** *classname*] ...
[**-**] *filename* [*filename*] ...

总是拒绝对这些文件的检索。如果文件采用绝对路径指定方式（即，以 **/** 字符开头），则只将这些文件标记为无法检索。否则，将拒绝传输具有匹配文件名的所有文件。例如：

noretrieve /etc/passwd core

指定任何人都无法获取 **/etc/passwd** 文件，但是，如果 **passwd** 文件不在 **/etc** 中，则允许他们传输该文件。另一方面，无论名为 **core** 的文件位于何处，任何人都无法获取它。

目录指定会将命名目录中的所有文件和子目录都标记为“无法获取”或不可获得。 *filename* 可指定为文件的文件名匹配替换形式。例如：

noretrieve /etc /home/*/.htaccess

指定不能检索 **/etc** 或其任何子目录中的文件。同样，不能检索位于 **/home** 目录下任何位置中的名为 **.htaccess** 的文件。

第一个可选参数用来选择名称被解释为指向由 **chroot** 更改到的当前环境的绝对路径还是相对路径。缺省值是将名称解释为以斜线开头的绝对路径。

可以对特定类的成员设置 **noretrieve** 限制。如果指定了任何 **class=** ，则只对特定类的用户设置该选项。

allow-retrieve [**absolute|relative**] [**class=** *classname*] ...
[**-**] *filename* [*filename*] ...

允许检索被 noretrieve 拒绝的文件。

loginfails *number*

在登录失败 *number* 次之后，记录 repeated**login**failures"" 消息并终止 FTP 连接。缺省值是 5。

private { **yes|no** }

用户登录后，可以使用 **SITE GROUP** 和 **SITE GPASS ftpd** 命令指定增强的访问组和相关的口令。如果组名和口令有效，则用户将（通过 **setgid()** ）成为组访问文件 **/etc/ftpd/ftpgroups** 中指定的组的成员。

组访问文件的格式如下：

access_group_name:*encrypted_password*:*real_group_name*

其中， *access_group_name* 是任意（字母数字和标点符号）字符串。 *encrypted_password* 是通过 **crypt**()（请参阅 *crypt*(3C) ）加密的口令，与 **/etc/passwd** 中的工作方式完全相同。 *real_group_name* 是 **/etc/group** 中列出的有效组的名称。

注释：为了使该选项适用于匿名 FTP 用户， FTP 服务器必须使 **/etc/group** 永久保持打开状态，并将组访问文件加载到内存中。这意味着：(1) 此时在 FTP 服务器上打开了另外一个文件描述符； (2) 通过 **SITE GROUP** （请参阅 *ftpd*(1M) ）授予用户的必需的口令和访问权限将在 FTP 会话期间保持静态。如果急需现在（立即）更改访问组和（或）口令，只需终止所有正在运行的 FTP 服务器。

信息功能

greeting { **full|brief|terse** }

greeting text *message*

允许控制在远程用户登录之前所提供的信息量。 **greeting full** 是缺省值，它显示主机名和守护程序版本。 **greeting brief** 显示主机名。 **greeting terse** 仅显示“FTP server ready”消息。此外，该消息将输出为 **STAT** 命令的输出。尽管 **full** 是缺省值，但是建议使用 **brief** 。

注释：不支持 **suppresshostname** 和 **suppressversion** 这两个选项。greeting 选项可用于禁止输出主机名或守护程序版本。

使用 **greeting text** *message* 形式，可以指定所需的任何问候消息。 *message* 可以是任何字符串；空白字符（空格和制表符）将转换为单个空格。

banner *path*

其工作方式与 **message** 命令相似（请参阅下文），不同之处是在用户输入用户名和口令之前显示横幅。 *path* 相对于实际系统根目录（而非匿名 FTP 目录的基址）。

警告：使用该命令可以完全禁止不兼容的 FTP 客户端使用 FTP 服务器。并非所有的客户端都可以处理多行响应（这是横幅的显示方式）。

hostname *some.host.name*

定义 FTP 服务器的缺省主机名。该字符串将在每次使用 **%L** Magic Cookie 时在问候消息中输出。有关 Magic Cookie 的列表，请参阅下面的 **message** 。该值将被虚拟服务器的主机名覆盖。如果未指定该值，则将使用本地计算机的缺省主机名。

email *name*

定义 FTP 归档维护人员的电子邮件地址。该字符串将在每次使用 **%E** Magic Cookie 时输出。有关 Magic Cookie 的列表，请参阅下面的 **message** 。

message *path* [*when* [*class*...]]

定义一个具有 *path* 的文件，以便 **ftpd** 在用户登录时或使用工作目录更改命令时向用户显示该文件的内容。 *when* 参数可以是 **LOGIN** 或 **CWD=***dir* 。如果 *when* 是 **CWD=***dir* ，则 *dir* 指定将触发通知的新缺省目录。

指定可选的 *class* 允许将消息仅显示给特定类的成员。可以指定多个类。

在消息文件中，用户可以键入消息并使用可用的“宏”或“Magic Cookie”。FTP 服务器将用指定的文本字符串替换 Magic Cookie。可使用下列 Magic Cookie：

%T 本地时间（形式为 Thu Nov 15 17:12:42 1990）。

%C 当前的工作目录

%E 在 ftpaccess 中定义的维护人员的电子邮件地址

%R 远程主机名

%L 本地主机名

%u 通过 RFC931 验证确定的用户名

%U 在登录时给出的用户名

%M 该类中允许的最大用户数量

%N 该类中当前的用户数

%B 对分配的磁盘块的绝对限制

%b 磁盘块的首选限制

%Q 当前的块数

%I 已分配的 i 节点的最大数量 (+1)

%i i 节点的首选限制

%q 已分配的 i 节点的当前数量

%H 磁盘使用率超额的时间限制

%h 文件比率超额的时间限制：

%xu 上载的字节数

%xd 下载的字节数

%xR 上载/下载比率 (1:n)

%xc 存储字节数

%xT 时间限制（分钟）

%xE 自登录以来经过的时间（分钟）

%xL 剩余的时间

%xU 上载限制

%xD 下载限制

该消息将只显示一次，以免打扰用户。请记住，当该消息由匿名 FTP 用户触发时， *path* 必须相对于匿名 FTP 目录树的基址。

readme *path* [*when* [*class*]]

定义一个具有 *path* 的文件，以便 **ftpd** 在用户登录时或使用工作目录更改命令时通知用户，该文件存在并且在某日期进行过修改。 *when* 参数可以是 **LOGIN** 或 **CWD=<dir>** 。如果 *when* 是 **CWD=<dir>** ，则 *dir* 指定将触发通知的新缺省目录。该消息将只显示一次，以免打扰用户。请记住，当 **README** 消息由匿名 FTP 用户触发时， *path* 必须相对于匿名 FTP 目录树的基址。

指定可选的 *class* 允许将消息仅显示给特定类的成员。可以指定多个类。

日志记录功能

log commands *typelist*

按用户启用各个命令的日志记录功能。 *typelist* 是以下任何关键字的逗号分隔列表： **anonymous** 、 **guest** 和 **real** 。如果包括 **real** 关键字，则将针对使用 FTP 访问实际帐户的用户执行日志记录功能；如果包括 **anonymous** 关键字，则将针对使用匿名 FTP 的用户执行日志记录功能。 **guest** 关键字与来宾访问帐户相匹配（有关详细信息，请参阅上面的访问功能小节中的 **guestgroup** ）。

log transfers *typelist directions*

为实际用户或匿名 FTP 用户启用文件传输的日志记录功能。以服务器为 接收方的（传入）传输的日志记录功能与以服务器为 发送方的（出站）传输的日志记录功能可以分别启用。 *typelist* 是以下任何关键字的逗号分隔列表： **anonymous** 、 **guest** 和 **real** 。如果包括 **real** 关键字，则将针对使用 FTP 访问实际帐户的用户执行日志记录功能。如果包括 **anonymous** 关键字，则将针对使用匿名 FTP 的用户执行日志记录功能。 **guest** 关键字与来宾访问帐户相匹配（有关详细信息，请访问上面的访问功能小节中的 **guestgroup**）。 *directions* 是以下两个关键字中任意关键字的逗号分隔列表： **inbound** 和 **outbound** 。第一个关键字将导致记录发送到服务器的文件的传输情况，第二个关键字将导致记录从服务器发送的文件的传输情况。所有的日志记录都写入 **/var/adm/syslog/xferlog** 文件中。

log security *typelist*

为实际用户、来宾用户和（或）匿名用户启用违反安全规则（noretrieve、.notar、...）的日志记录。 *typelist* 是以下任何关键字的逗号分隔列表： **anonymous** 、 **guest** 和 **real** 。如果包括 **real** 关键字，则将针对使用 FTP 访问实际帐户的用户执行日志记录功能。如果包括 **anonymous** 关键字，则将针对使用匿名 FTP 的用户执行日志记录功能。 **guest** 关键字与来宾访问帐户相匹配（有关详细信息，请参阅 **guestgroup**）。

log syslog

log syslog+xferlog

将传入和传出传输的日志记录消息重定向到 **syslog** 或 **xferlog** 或者重定向到二者。缺省情况下（如果未指定 **log** ），会将传输日志消息放到 **xferlog** 中。 **log syslog** 仅将日志消息放到 **syslog** 中。 **log syslog+xferlog** 会将日志消息同时放到 **syslog** 和 **xferlog** 中。

上载/下载比率

ul-dl-rate *rate* [*class* ...]

指定上载/下载比率 (1:*rate*) 。对于 FTP 用户上载的每个字节，可以下载比率为 *rate* 的字节。缺省情况下没有任何比率。

dl-free *filename* [*class* ...]

可以忽略上载/下载比率自由地下载 *filename* 文件。请参阅上面的 **ul-dl-rate** 。

dl-free-dir *dirname* [*class* ...]

可以忽略上载/下载比率自由地下载 *dirname* 目录及其子目录中的所有文件。请参阅上面的 **ul-dl-rate** 。

请注意， **dl-free** 和 **dl-free-dir** 均相对于系统的根目录环境（而非 **chroot** 环境）。

其他功能

alias *string dir*

为指定的目录 *dir* 定义别名 *string* 。可用于添加逻辑目录这一概念。

例如：

alias rfc /pub/doc/rfc

将允许用户从任何目录通过 **cd rfc:** 命令来访问 **/pub/doc/rfc** 。别名仅适用于 **cd** 命令。

cdpath *dir*

在 **cdpath** 中定义一个目录条目。 *dir* 定义一个在更改目录时所使用的搜索路径。

例如：

cdpath /pub/packages

cdpath /.aliases

将允许用户执行 **cd** 以进入直接位于 **/pub/packages** 或 **/.aliases** 目录下的任何目录。搜索路径按照相应的行在 **/etc/ftpd/ftpaccess** 文件中出现的顺序来定义。

如果用户给出以下命令：

cd foo

则将按照以下顺序来搜索该目录：

./foo 名为 **foo**
的别名.B /pub/packages/foo
/.aliases/foo

cd 路径仅适用于 **cd** 命令。如果有大量别名，则可能需要设置一个别名目录，并在其中包含一些链接，让它们指向要供用户使用的所有区域。

compress { **yes|no** } *classglob* [*classglob* ...]
tar { **yes|no** } *classglob* [*classglob* ...]

对于与任一 *classglob* 相匹配的任何类，启用 **compress** 或 **tar** 功能。实际的转换是在外部文件 **/etc/ftpd/ftpconversions** 中定义的。

shutdown *path*

如果 *path* 所指向的文件存在，则服务器将定期检查该文件，以确定服务器是否要关闭。如果已计划关闭服务器，则用户会得到通知，并且在关闭之前的指定时间内拒绝新连接，同时，在关闭之前的指定时间断开当前连接。 *path* 指向具有如下结构的文件：

year month day hour minute deny_offset disc_offset
text

year	> 1970 的任何年份
month	**0-11** <-- 注释：从 **0** 开始的月份索引
hour	**0-23**
minute	**0-59**

deny_offset 和 *disc_offset* 是 *HHMM* 格式的偏移量，前者表示在关闭服务器之前拒绝新连接的时间偏移量，后者表示在关闭服务器之前断开现有连接的时间偏移量。

text 遵循任何消息的标准规则（请参阅“信息功能”小节中的 **message** ），还可以使用下列附加的 Magic Cookie：

%s 系统将要关闭的时间
%r 将拒绝新连接的时间
%d 将断开当前连接的时间

所有的时间都采用如下形式： *ddd MMM DD hh:mm:ss YYYY* 。在配置文件中只能有一个 **shutdown** 命令。

可以使用外部程序 **ftpshut** 自动执行该文件的生成过程。

daemonaddress *address*

如果未设置该值，则服务器将在每个 IP 地址上监听连接。否则，它将只在指定 IP 地址上监听。因为该子句会中断虚拟宿主连接，所以不建议使用它。仅当 **ftpd** 以独立模式运行时（请参阅 *ftpd*(1M)），该选项才起作用。

virtual *address* { **root|banner|logfile** } *path*

启用虚拟 FTP 服务器功能。 *address* 是虚拟服务器的 IP 地址。第二个参数指定 *path* 是下列值之一:

root 该虚拟服务器的文件系统的根目录。

banner 当用户连接到该虚拟服务器时向用户显示的横幅。

logfile 在其中记录该虚拟服务器的传输情况的日志文件。如果未指定 **logfile**，则将使用缺省日志文件。

所有其他消息文件和权限以及此文件中的任何其他设置都适用于所有的虚拟服务器。

也可以将 *address* 指定为主机名（而非 IP 编号）。但是强烈建议不要这样做，这是因为：如果 DNS 在 FTP 会话开始时不可用，则主机名将不匹配。

virtual *address* { *hostname*|*email* } *string*

将 *string* 设置为问候消息和 **STAT** 命令中所显示的主机名，或设置为消息文件和 **HELP** 命令中使用的电子邮件地址。

virtual *address* **allow** *username* [*username* ...]
virtual *address* **deny** *username* [*username* ...]

通常情况下，不允许实际用户和来宾用户登录虚拟服务器，但他们是来宾而且已通过执行 **chroot** 将根目录更改到虚拟根目录时除外。列在 **virtual allow** 行上的用户将被授予访问权限。通过将“*”用作用户名，可以向所有的用户授予访问权限。 **virtual deny** 子句是在 **virtual allow** 子句之后处理的，它可用于在允许所有用户访问时拒绝特定用户访问。

virtual *address* **private**

通常情况下，允许匿名用户登录虚拟服务器。该选项拒绝匿名用户进行访问。

virtual *address* **passwd** *file*

为虚拟域使用其他 passwd *file* 。

注释：HP-UX 当前不支持该选项。

virtual *address* **shadow** *file*

为该虚拟域使用其他 shadow *file* 。

注释：HP-UX 当前不支持该选项。

defaultserver deny *username* [*username* ...]
defaultserver allow *username* [*username* ...]

通常情况下，允许所有用户访问缺省（非虚拟）FTP 服务器。使用 **defaultserver deny** 可以撤消特定用户的访问权限。指定 **defaultserver deny** 可以拒绝所有用户的访问。然后，可以使用 **defaultserver allow** 允许特定用户访问。

defaultserver private

通常情况下，允许匿名用户访问缺省（非虚拟）FTP 服务器。该语句不允许进行匿名访问。

virtual 、 **defaultserver allow** 、 **deny** 和 **private** 子句提供了一种控制允许哪些用户访问哪个 FTP 服务器的方法。

passive address *externalip cidr*

允许控制为了响应 **PASV** 命令而报告的地址。当与 *cidr* 相匹配的任何控制连接请求被动数据连接 (**PASV**) 时，将报告 *externalip* 地址。

注释：这不会更改守护程序实际监听的地址，只是向客户端报告地址。该功能允许守护程序在 IP 重新编号防火墙后面正确地运行。例如：

passive address 10.0.1.15 10.0.0.0/8
passive address 192.168.1.5 0.0.0.0/0

从 A 类网络 10 建立连接的客户端将被告知正在 IP 地址 10.0.1.15 上监听被动连接，而所有其他客户端将被告知正在 192.168.1.5 上监听连接。

可以指定多个被动地址以处理复杂网络或多网关网络。

注释： *IPv6 enabled systems* 不支持该选项。

passive ports *cidr min max*

允许控制可用于被动数据连接的 TCP 端口号。如果控制连接与 *cidr* 相匹配，则在 *min* 至 *max* 范围内为守护程序随机选择一个要监听的端口。该功能允许防火墙限制远程客户端可用于连接到受保护网络的端口。

cidr 是 IP 地址的简写形式，它采用点分形式，后跟一个斜线和代表网络地址的最左边位数（与计算机地址相对）。例如，如果使用保留的 A 类网络 10 （而非网络掩码 255.0.0.0），则在 10.0.0.0/8 中用 *cidr* 为 /8 来代表您的网络。

注释： *IPv6 enabled systems* 不支持该选项。

pasv-allow *class* [*addrglob* ...]
port-allow *class* [*addrglob* ...]

通常情况下，守护程序不允许 **PORT** 命令指定控制连接地址以外的地址。而且它不允许从另一个地址建立 **PASV** 连接。

port-allow 子句提供一个地址列表，指定 *class* 的用户可以在 **PORT** 命令中给出这些地址。即使这些地址与控制连接的客户端的 IP 地址不匹配，也将允许使用这些地址。

pasv-allow 子句提供一个地址列表，指定 *class* 的用户可以从这些地址建立数据连接。即使这些地址与控制连接的客户端的 IP 地址不匹配，也将允许使用这些地址。

lslong *command* [*options* ...]
lsshort *command* [*options* ...]
lsplain *command* [*options* ...]

lslong 、 **lsshort** 和 **lsplain** 子句指定用来生成目录列表的命令和命令选项。请注意，这些选项不能包含空格。通常， **/usr/bin/ls** 命令用于提供目录列表。要更改 **ls** 的路径，请在 **command** 中指定它。这些子句的缺省值通常是正确的。对于普通用户，将使用 **lsshort** 。对于匿名用户，将使用 **lslong** 。对于特殊情况，将使用 **lsplain** 。仅当绝对必要时，才使用 **lslong** 、 **lsshort** 或 **lsplain** 。

mailserver *hostname* [*hostname* ...]

指定将接受 FTP 守护程序的上载通知的邮件服务器的名称。可以列出多个邮件服务器；守护程序将尝试按顺序向每个服务器发送上载通知，直到一个服务器接受该消息为止。如果未指定邮件服务器，则将使用本地主机。仅当有人得到匿名上载通知时（请参阅下面的 **incmail** ），该选项才有意义。

incmail *emailaddress*
virtual *address* **incmail** *emailaddress*
defaultserver incmail "*emailaddress"*

指定要得到匿名上载通知的电子邮件地址。可以指定多个地址；每个地址将接收一个通知。如果未指定任何地址，将不发送通知。

如果为 **virtual** 主机指定了地址，则只有这些地址将接收该主机上的匿名上载通知。否则，将向全局地址发送通知。

defaultserver 地址仅适用于实际主机，而不适用于虚拟主机。这样，实际主机可以接收其缺省匿名区域的上载通知。但是，如果设置了该选项，则将不通知虚拟主机。

mailfrom *emailaddress*
virtual *address* **mailfrom** *emailaddress*
defaultserver mailfrom *emailaddress*

指定匿名上载通知的发件人电子邮件地址。只能指定一个地址。如果未应用 **mailfrom** ，则将从缺省邮箱名“wu-ftpd”发送电子邮件。为避免在收件人尝试回复通知或者在下行邮件问题生成退回邮件时出现问题，则应当确保 **mailfrom** *emailaddress* 可以传递。

权限功能

chmod { **yes|no** } *typelist*

delete { **yes|no** } *typelist*

overwrite { **yes|no** } *typelist*

rename { **yes|no** } *typelist*

umask { **yes|no** } *typelist*

允许或禁止执行指定功能的能力。缺省情况下，允许所有用户执行。

typelist 是以下任意关键字的逗号分隔列表： **anonymous** 、 **guest** 、 **real** 和 **class=** 。当 **class=** 出现时，它后面必须跟着一个类名。 **class=** ，则 *typelist* 限制仅适用于该类用户。

passwd-check { **none|trivial|rfc822** } [**enforce|warn**]

定义服务器为匿名 FTP 执行口令检查的级别和强制性。

none 不执行口令检查。

trivial 口令中必须包含 @ 。

rfc822 口令中必须包含符合 rfc822 的地址。

warn 警告用户，但是允许他们登录。

enforce 警告用户，然后将他们注销。

deny-email *case-insensitive-email-address*

以参数形式提供的电子邮件地址将被视为无效。如果将 **passwd-check** 设置为 enforce，则以口令形式提供该地址的匿名用户不能登录。通过这一方法，可以禁止用户在 Web 浏览器中使用虚设地址（如 IE?0User@ 或 mozilla@）。通过使用 **deny-email** ，不会将使用 Web 浏览器访问 FTP 的用户排除在外。只需让他们正确地配置自己的浏览器。每行只有一个地址，但是可以根据需要包含任意多 **deny-email** 子句。

path-filter *typelist mesg allowed_charset* [*disallowed_regexp* ...]

对于 *typelist* 中的用户， **path-filter** 定义用来控制文件名可以是什么或者不能是什么的正则表达式。可以用多个正则表达式（请参阅 *regexp*(5)）来指定不允许使用的正则表达式 *disallowed_regexp* 。如果文件名由于无法与正则表达式条件相匹配而无效，则将向用户显示 *mesg* 。例如：

path-filter anonymous /etc/pathmsg ^[-A-Za-z0-9_\.]*$ ^\. ^-

指定匿名用户的所有上载文件名必须只包含以下字符： **A-Z** 、 **a-z** 、 **0-9** 、句点 (**.**) 、短线 (**-**) 和下划线 (_) 。文件名不能以句点 (**.**) 或短线 (**-**) 开头，它们分别用 ^\. 和 ^- 来指定。如果文件名无效，则将向用户显示 **/etc/pathmsg** 。

upload [**absolute|relative**] [**class=** *classname*]... [-] *root-dir dirglob* { **yes|no** } *owner group mode* [**dirs|nodirs**] [*d_mode*]

定义一个其 *dirglob* 允许或拒绝上载的目录。

如果它确实允许上载，则新创建的所有文件将由 *owner* 和 *group* 拥有，其权限根据 *mode* 进行设置。被覆盖的现有文件将保持其最初的所有权和权限。

目录将在最佳匹配基础上进行匹配。

例如：

upload /var/ftp * no
upload /var/ftp /incoming yes ftp daemon 0666
upload /var/ftp /incoming/gifs yes jlc guest 0600 nodirs

这些 **upload** 命令将只允许上载到 **/incoming** 和 **/incoming/gifs** 。上载到 **/incoming** 的文件将由 **ftp/daemon** 拥有，而且将具有 **0666** 权限。上载到 **/incoming/gifs** 的文件将由 **jlc/guest** 拥有，而且将具有 **0600** 权限。请注意，此处的 *root-dir* 必须与口令数据库中为 **ftp** 用户指定的主目录相匹配。

可以指定可选的 **dirs** 和 **nodirs** 关键字，以便允许或禁止通过 **mkdir** 命令创建新的子目录。

请注意，如果使用 **upload** 命令，则缺省情况下将允许创建目录。要缺省禁用该功能，必须指定用户、组和模式并后跟 **nodirs** 关键字，将其作为在该文件中使用 **upload** 命令的第一行。

如果允许创建目录，则可选的 *d_mode* 确定新建目录的权限。如果省略了 *d_mode* ，则将从 *mode* 推断权限，或者如果同时省略了 *mode* ，则权限将为 **0777** 。

upload 仅适用于拥有主目录 *root-dir* （chroot() 的参数）的用户。可以将 *root-dir* 指定为"*"以匹配所有主目录。

owner 和（或） *group* 可以分别指定为"*"，这种情况下，任何已上载的文件或目录将用在其中创建它们的目录的所有权来创建。

第一个可选参数用来选择 *root-dir* 名称被解释为指向由 **chroot** 更改到的当前环境的绝对路径还是相对路径。缺省值是将 *root-dir* 名称解释为绝对路径。

可以指定任意数量的 **class=***classname* 限制。如果指定了任何限制，则只有在当前用户是其中某一类的成员时，该 upload 子句才生效。

anonymous-root *root-dir* [*class* ...]

root-dir 为匿名用户指定 **chroot()** 路径。如果没有匹配的 **anonymous-root** ，则将使用对 FTP 用户的主目录进行解析的旧方法。如果未指定 *class* ，则 *root-dir* 是没有任何其他 **anonymous-root** 指定的匿名用户的根目录。可以在该行上提供多个类。如果为用户选择了 **anonymous-root** ，则使用 *root-dir***/etc/passwd** 文件中 FTP 用户的主目录来确定初始目录，不使用系统级 **/etc/passwd** 中 FTP 用户的主目录。例如：

anonymous-root /home/ftp
anonymous-root /home/localftp localnet

会导致所有匿名用户通过执行 **chroot()** 更改到 **/home/ftp** 目录。然后，如果 FTP 用户存在于 **/home/ftp/etc/passwd** 中，则他们的初始 **CWD** 就是主目录。但是，localnet 类的匿名用户将通过执行 **chroot()** 更改到 **/home/localftp** 目录，他们的初始 **CWD** 将从 **/home/localftp/etc/passwd** 中 FTP 用户的主

目录提取。

guest-root *root-dir* [*uid-range* ...]

root-dir 为来宾用户指定 **chroot()** 路径。如果 **guest-root** 不匹配，则将使用对用户的主目录进行解析的旧方法。如果未指定 *uid-range* ，则根目录适用于不与任何其他 *guest-root* 指定匹配的来宾用户。可以在该行上提供多个 UID 范围。如果为用户选择了 **guest-root** ，则将使用 *root-dir/etc/passwd* 文件中用户的主目录来确定初始目录，不使用系统级 **/etc/passwd** 中用户的主目录。

uid-range 指定数字 UID 值。范围可通过给出下限和上限（包括下限和上限）、中间用短线分隔来指定。省略下限表示“小于某个值的所有值”，省略上限表示“大于某个值的所有值”。例如：

guest-root /home/users guest-root /home/staff %100-999 sally

guest-root /home/users/frank/ftp frank

会导致所有来宾用户执行 **chroot()** 以更改到 **/home/users** ，然后在 **/home/users/etc/passwd** 中所指定的主目录中启动每个用户。介于 100 和 999 之间的用户（包括 100 和 999）以及 **sally** 用户将通过执行 **chroot()** 更改到 **/home/staff** ，而且 CWD 将从他们在 **/home/staff/etc/passwd** 中的条目提取。单个用户 **frank** 将通过执行 **chroot()** 更改到 **/home/users/owner/ftp** ，而且 CWD 将从他在 **/home/users/owner/ftp/etc/passwd** 中的条目提取。

请注意，对于 **anonymous-root** 和 **guest-root** 来说，顺序至关重要。如果用户与多个子句相匹配，则只应用第一个子句；但是不含 *class* 或 *uid-range* 的子句除外，它们仅在没有其他匹配子句时才应用。

deny-uid *uid-range* [...]

deny-gid *gid-range* [...]

allow-uid *uid-range* [...]

allow-gid *gid-range* [...]

这些子句允许指定将被拒绝访问 FTP 服务器的 UID 和 GID 值。对于被拒绝的 UID（或 GID），可以使用 **allow-uid** 和 **allow-gid** 子句允许它们进行访问。这些检查将在所有其他检查之前进行。先检查拒绝权限再检查允许权限。缺省值是允许访问。请注意，大多数情况下，这可以消除对 **/etc/ftpd/ftpusers** 文件的需要。例如：

deny-gid %-99 %65535 deny-uid %-99 %65535

allow-gid ftp

allow-uid ftp

拒绝 Linux 终端上所有的特权用户和组或特殊用户和组进行 FTP 访问，但匿名 FTP 用户（或组）除外。在许多情况下，这可以消除对 **/etc/ftpd/ftpusers** 文件的需要。系统仍然支持该文件，以便在不希望更改 **/etc/ftpd/ftpaccess** 时使用它。

在整个 **ftpaccess** 文件中，可以在允许使用单个 UID 或 GID 的任何位置使用名称或数字。要使用数字，请在它前面放一个 **%** 。在允许使用某个范围的位置，在该范围之前放一个 **%** 。

restricted-uid *uid-range* [...]

restricted-gid *gid-range* [...]
unrestricted-uid *uid-range* [...]
unrestricted-gid *gid-range* [...]

这些子句控制是否允许实际用户或来宾用户访问 FTP 站点上其主目录以外的区域。这些子句并不旨在取代 guestgroup 和 guestuser 的使用。相反，使用它们可以补充来宾操作。使用 **unrestricted-uid** 和 **unrestricted-gid** 子句，可以允许被限制访问其主目录之外区域的用户访问这些区域。

这些子句的使用示例可以说明它们的用途。假设用户 **dick** 的主目录为 **/home/dick** ，用户 **jane** 的主目录为 **/home/jane** ：

guest-root /home dick jane
restricted-uid dick jane

虽然 **dick** 和 **jane** 都可通过执行 **chroot** 更改到 **/home** ，但是他们不能互相访问对方的文件，因为他们被限制在各自的主目录。

在该示例等情况下，如有可能，请尝试不要单独依赖于 FTP 限制。对于所有其他 FTP 访问规则，请尝试使用目录和文件权限来防止对 ftpaccess 配置执行操作。

注释：对于上述子句，必须将 **/usr/lib/libnss_files.1** 和 **/usr/lib/libdld.2** 库复制到通过执行 **chroot** 更改到的当前环境的 **/usr/lib** 目录。

site-exec-max-lines *number* [*class* ...]

传统上，SITE EXEC 功能限制可以发送到远程客户端的输出的行数。该子句允许设置这一限制。如果省略该参数，则限制是 20 行。如果限制是 0（零），则表示没有限制。如果选择删除该限制，则一定要格外小心。如果发现某个子句与远程用户的类相匹配，则将使用该限制。否则，将使用具有“*”类或者没有指定类的子句。例如：

site-exec-max-lines 200 remote
site-exec-max-lines 0 local
site-exec-max-lines 25

在上述示例中，对于 **remote** 用户，将 SITE EXEC（因之将执行 SITE INDEX）的输出限制为 **200** 行；对于 **local** 用户，指定没有任何限制；对于所有其他用户，将限制设置为 **25** 行。

dns refuse_mismatch *filename* [**override**]

当对远程站点的正向和反向查找不匹配时，将拒绝执行 FTP 会话。会显示命名文件 *filename* （如消息文件），并警告用户。如果指定了可选的 **override** ，则允许在输出警告后建立连接。

dns refuse_no_reverse *filename* [**override**]

当没有远程站点的反向 DNS 条目时，将拒绝执行 FTP 会话。会显示命名文件 *filename* （如消息文件），并警告用户。如果指定了可选的 **override** ，则允许在输出警告后建立连接。

dns resolveroptions [*options*]

dns resolveroptions 用于调整名称服务器选项。该行可以采用一系列标志（删除了前导 RES_），这些标志在 *resolver*(3N) 中进行了说明。每个标志前面可以有一个可选的 + 或 -。例如，

dns resolveroptions +aaonly -dnsrch

将启用 **aaonly** 选项（仅接受授权答复）并禁用 **dnsrch** 选项（搜索域路径）。

警告

HP-UX 11i v1.0 支持 IPv6，需要安装可选的 IPv6 软件。目前，运行 HP-UX 11i v1.6 的系统不支持 IPv6。

文件

/etc/ftpd/ftpaccess

作者

ftpaccess 由密苏里州华盛顿大学圣路易斯分校开发。

另请参阅

ftpshut(1)、 groups(1)、 passwd(1)、 ftpd(1M)、 chroot(2)、 umask(2)、 resolver(3N)、 ftpconversions(4)、 ftpgroups(4)。

名称

ftpconversions - ftpd 转换数据库

概要

/etc/ftpd/ftpconversions

说明

ftpd 所知的转换及其属性保存在一个 ASCII 文件中，该文件构建如下。该文件中的每一行描述一个 转换。字段以冒号 (:) 分隔。

%s:%s:%s:%s:%s:%s:%s:%s
1 2 3 4 5 6 7 8

字段	说明
1	strip prefix
2	strip postfix
3	addon prefix
4	addon postfix
5	external command
6	types
7	options
8	description

strip postfix 和 *addon postfix* 可以是文件扩展名 **.Z** 、 **.tar** 、 **.gz** 或 **.tar**[**.Z**,**.gz**] 。该文件扩展了 ftp 服务器的支持并将 ftp 服务器执行的操作汇总如下表所示。该功能只在使用 **get** 命令时被支持。

真实文件名	指定文件名	操作
filename.Z	*filename*	发送前解压缩文件
filename	*filename*.Z	发送前压缩 *filename*
filename	*filename*.tar	发送前将 *filename* 转换为 tar 文件
filename	*filename*.tar.Z	将 *filename* 转换为 tar 文件并压缩发送前

external command 字段用于指定执行上表中描述的不同文件扩展名操作的程序或脚本路径。

types 字段表示在 **get** 命令要求的文件进行压缩、解压缩、转换为 tar 文件及转换为非 tar 文件之前必须选中的标志。

options 字段指定了 *external command* 字段中指定的程序或脚本选项。

description 是每个转换的简短描述。

注释：HP-UX **tar** 不支持进行 **tar+compress** 或 **tar+gzip** 的任何选项。如果用户想执行任何那样的操作，则他/她必须编写他/她自己的程序或脚本来完成并在 **/etc/ftpd/ftpconversions** 文件中的 *external command* 字段中指定该程序或脚本的路径名。用于执行转换的程序/脚本的副本及该程序/脚本各自使用的库也必须复制到每个 **anonymous**

和 **guest** 用户帐户来使 **anonymous** 和 **guest** 用户可以执行希望的转换。

警告

该转换机制目前不支持 *strip prefix* 和 *addon prefix* 字段。

文件

etc/ftpd/ftpconversions

作者

ftpconversions 由密苏里州华盛顿大学圣路易斯分校开发。

另请参阅

ftpd(1M)、 ftpaccess(4)。

名称

ftpgroups - 与 SITE GROUP 和 SITE GPASS 命令一起使用的组口令文件。

概要

/etc/ftpd/ftpgroups

说明

文件 **ftpgroups** 是与 SITE GROUP 和 SITE GPASS 命令一起使用的组口令文件。

若要使用该文件，必须在配置文件 **/etc/ftpd/ftpaccess** 中建立 **private yes** 条目。

该文件包含了一个字母数字型字符串、加密口令以及来自文件 **/etc/group** 的真实组名。每个条目以 **:** 分隔。当用户登录后，SITE GROUP 和 SITE GPASS 命令可用于指定增强的访问组及关联的口令。如果组名和口令有效，则用户成为组访问文件 **/etc/ftpd/ftpgroups** 中指定组的成员（通过 **setgid()** ）。

组访问文件的格式如下:

access_group_name:*encrypted_password*:*real_group_name*

其中， *access_group_name* 是个任意字符串（字母数字和（或）标点符号）。 *encrypted_password* 是通过 *crypt*(3) 加密的口令，与 **/etc/passwd** 中非常相似。 *real_group_name* 是 **/etc/group** 中列出的有效组的名称。

注释：对于这个为匿名 FTP 用户服务的选项，ftp 服务器必须保持 **/etc/group** 永久打开，并且组访问文件会加载到内存。这意味着:

(1) 当前ftp 服务器将打开附加的文件描述符，并且

(2) 必需的口令和通过 SITE GROUP 授予用户的访问权限，在一个 FTP 会话期间将是静态的。

如果用户急需更改访问组和（或）口令 *NOW*，则应终止所有运行中的 FTP 服务器。

文件

/etc/ftpd/ftpgroups

作者

ftpgroups 由密苏里州华盛顿大学圣路易斯分校开发。

另请参阅

ftpaccess(4)。

名称

ftphosts - ftpd 单用户主机访问文件

概要

/etc/ftpd/ftphosts

说明

/etc/ftpd/ftphosts 文件用于允许或禁止不同主机对特定帐户的访问。

访问功能

allow *username addrglob* [*addrglob*...]

仅允许与 *addrglob* 相匹配的主机作为 *username* 登录。 **addrglob** 是全局域名或全局数字形式的地址。

deny *username addrglob* [*addrglob* ...]

总是拒绝匹配 *addrglob* 的主机作为 *username* 登录。

anonymous 用户名或 **ftp** 指定匿名用户。

addrglob 也可以指定为“地址/cidr”或“地址:netmask”。例如:

10.0.0.0/8 或 10.0.0.0:255.0.0.0

举例

allow bartm some.domain 131.211.32.*

deny fred other.domain 135.112.23.*

allow sam any.domain

文件

/etc/ftpd/ftphosts

作者

ftphosts 由密苏里州华盛顿大学圣路易斯分校开发。

另请参阅

ftpd(1M)、 ftpaccess(4)、 ftpconversions(4)、 ftpshut(1)。

名称

ftpservers - ftpd 虚拟主机配置规格说明文件

概要

/etc/ftpd/ftpservers

说明

/etc/ftpd/ftpservers 文件用于通知 **ftpd** 服务器应使用哪一个虚拟域配置文件集。在 **VIRTUAL** 支持下，wu-ftpd 可以为每一个虚拟域使用单独的配置文件。

对于一个虚拟主机，配置文件可放置到一个单独的虚拟域目录中。目录路径和使用那些配置文件的虚拟主机 IP 地址都列在 **/etc/ftpd/ftpservers** 文件中。放入虚拟域目录的实际配置文件必须以如下方式命名:

ftpaccess 虚拟域的访问文件。

ftpusers 限制可使用该 Web 服务器的帐户。

ftpgroups SITE GROUP 和 SITE GPASS 支持。

ftphosts 允许或拒绝用户名访问该虚拟服务器。

ftpconversions 自定义虚拟域中的转换。

无须将每个文件都放入每一个虚拟域目录中。如果想使一个域集使用一个文件的主副本，例如， **ftpconversions** 文件，则不要将这个文件包括在那些虚拟域目录中。这种情况下，将使用缺省的主副本。

注释：文件名必须与上述列出的名称相匹配。如果拼错任意名称或将它们命名为其他名称，则服务器将查找不到这些名称，并将使用它们的主副本来替代。

ftpservers 文件格式

每一个条目都有两个字段。

ipaddr/hostname directory-containing-configuration-files

例如:

10.196.145.10 /etc/ftpd/ftpaccess.somedomain/

10.196.145.200 /etc/ftpd/ftpaccess.someotherdomain/

some.domain INTERNAL

当一个 ftp 客户端连接到服务器时， **ftpd** 尝试将该 ftp 客户端所连接的 IP 地址与 **ftpservers** 文件中的一个地址进行匹配。

如果找到匹配项，则返回指定目录的路径，且该目录中包含该特定域的配置文件。 **ftpd** 服务器使用该目录中的任意配置文件。

如果未找到匹配项，或遇到无效的目录路径，则使用缺省的配置文件路径。上例中使用 **INTERNAL** 检查特定目录失败，并将使用主配置文件。

可使用实际 IP 地址或特定的主机名来指定虚拟域。较好的是指定虚拟域的实际 IP，因为这样可减少对域查找的

需求，并可消除 DNS 安全性相关的命名问题。

注释：注释行由一个 # 符号描述。忽略注释行和空行。

文件

/etc/ftpd/ftpservers

作者

ftpservers 由密苏里州华盛顿大学圣路易斯分校开发。

另请参阅

ftpd(1M)、 ftpaccess(4)、 ftpconversions(4)、 ftphosts(4)。

名称

ftpusers - ftpd 的安全文件

说明

ftpd 拒绝远程登录到 **/etc/ftpd/ftpusers** 中命名的本地用户帐户。每个受限帐户名在该文件中必须以单独一行出现。行不能包含任何空白字符。在 **/etc/passwd** 中指定了受限登录 Shell 的用户帐户必须在 **/etc/ftpd/ftpusers** 中列出，因为 **ftpd** 访问本地帐户，但并不使用它们的登录 Shell。 UUCP 帐户应在 **/etc/ftpd/ftpusers** 列出。如果没有 **/etc/ftpd/ftpusers** ， **ftpd** 将跳过安全检查。

举例

给定的 **/etc/ftpd/ftpusers** 文件包含以下内容：

```
Only lines that exactly match user account names are
significant.  Blank lines are harmless because they
do not match any account names.  However you must be
careful.

uucp
guest
```

ftpd 将拒绝使用本地帐户 **uucp** 或 **guest** 的登录尝试。

作者

ftpusers 由加州大学伯克利分校和密苏里州华盛顿大学圣路易斯分校联合开发。

另请参阅

ftp(1)、ftpd(1M)。

名称

gated.conf - GateDaemon 配置指南

概要

/etc/gated.conf

说明

配置指南概述

- 简介
- 语句摘要
- 优先级和路由选择
- 跟踪语句和全局选项
- 指令语句
- 选项语句
- 接口语句和配置
- 定义语句

协议语句

- 协议概述
- 内部网关协议 (igp)
    - RIP、HELLO、OSPF
- 外部网关协议 (egp)
    - EGP、BGP
- ICMP 语句
- 重定向语句
- 路由器发现语句
- 内核接口
- 静态路由

控制语句

- 路由过滤
- 匹配 AS 路径
- 路由导入

- 路由导出
- 路由汇聚

附录

- 术语表
- 参考资料

GateD 配置简介

语法

GateD 配置文件由一系列分号 (**;**) 分隔的语句组成。语句由空格分隔的标记组成，这些标记可以是空白字符、制表符和换行符的任意组合。该结构便于识别相互关联以及与特定协议相关联的配置部分。注释可以通过两种形式之一来指定。一种形式以井号 (**#**) 开头，一直延续到行尾。另一种形式是 *C* 形式，它以 **/*** 开头，一直延续到 ***/**。

语法说明惯例

分析程序所识别的确切关键字和特殊字符使用粗体字体显示。参数以斜体变量定义样式显示。方括号 (**[** 和 **]**) 中的参数用于显示可选的关键字和参数。竖线 (**|**) 用于指示在可选参数之间选择。括号 (**(** 和 **)**) 在必要时用于将关键字和参数分组。

例如，如下语法说明：

[**backbone** | (**area** *area*)]

方括号指示任一参数均是可选的。关键字是 **backbone** 和 **area**。竖线指示可以指定“**backbone** ”或“**area** *area*”。由于 *area* 采用变量定义样式，它是需要提供的参数。

语句分组

配置语句以及这些语句出现的顺序将 **gated.conf** 分为选项语句、接口语句、定义语句、协议语句、静态语句、控制语句和汇聚语句。如果不按顺序输入语句，在分析配置文件时将导致错误。

另外有两种语句不适合这些类别： %directive 语句和 %trace 语句。这些语句向分析程序提供指令，并控制配置文件中的跟踪。它们与任何协议的配置都无关，并且可以出现在 gated.conf 文件中的任意位置。

语句摘要

配置语句摘要表（在配置语句摘要中）按名称列出每个 GateD 配置语句，标识语句类型，并提供命令功能的简短概要。随后的各个部分将提供八个 GateD 语句类别中每一个的详细定义和说明。

GateD 配置语句摘要

GateD 配置命令摘要如下。该表按名称列出每个命令，标识语句类型，并提供语句功能的概要：

GateD 配置语句摘要

%directory (directive)	设置包含文件的目录。
%include (directive)	将一个文件包括到 gated.conf 中。

traceoptions (trace)	指定跟踪的事件。
options (definition)	定义 GateD 选项。
interfaces (definition)	定义 GateD 接口。
autonomoussystem (definition)	定义 AS 编号。
routerid (definition)	定义源路由器（BGP、OSPF）。
martians (definition)	定义无效的目标地址。
rip (protocol)	启用 RIP 协议。
hello (protocol)	启用 HELLO 协议。
isis (protocol)	启用 ISIS 协议。
kernel (protocol)	配置内核接口选项。
ospf (protocol)	启用 OSPF 协议。
egp (protocol)	启用 EGP 协议。
bgp (protocol)	启用 BGP 协议。
redirect (protocol)	配置 ICMP 重定向的处理。
icmp (protocol)	配置通用 ICMP 数据包的处理。
static (static)	定义静态路由。
import (control)	定义要导入的路由。
export (control)	定义要导出的路由。
aggregate (control)	定义要汇聚的路由。
generate (control)	定义要生成的路由。

优先级

优先级是 GateD 用于将来自一个协议或对等端的优先路由排列在其他路由之前的值。在 GateD 配置文件中可以用多个不同的配置语句来设置优先级。优先级可以基于网络接口的优先顺序、协议的优先顺序或远程网关的优先顺序来设置。优先级不得用于控制在 **igp** 中的路由选择，这由协议基于度量单位来自动完成。优先级可用于从来自不同对等端或自治系统的相同 **egp** 中选择路由。即使可以在配置文件中的多处设置优先级，每个路由也只有一个与之关联的优先级值。简单而言，为路由设置的最后一个或最具体的优先级值就是所使用的值（请参阅 术语表：优先级）。优先级值是任意分配的值，用于在单个路由数据库中确定到同一目标的路由的顺序。活动的路由按最低的优先级值来选择。某些协议实现另一种优先级 (preference2)，有时称作仲裁级。

选择路由

- 首选具有最佳（数字最小）优先级的路由。

- 如果两条路由具有相同的优先级，将首选具有最佳（数字最小）preference2（也称作仲裁级）的路由。

- 从 **igp** 得到的路由比从 **egp** 得到的路由具有更高的优先级。优先级最低的路由通过 **igp** 从 **egp** 中间接得到。
- 如果 AS 路径信息可用，则将其用于帮助确定优先级最高的路由。
  - 具有 AS 路径的路由其优先级高于没有 AS 路径的路由。
  - 如果 AS 路径和来源相同，则首选具有较小度量单位的路由。
  - AS 路径来源为 **igp** 的路由其优先级高于 AS 路径来源为 **egp** 的路由。优先级最低的是来源为 **unknown** 的 AS 路径。
  - 首选 AS 路径较短的路由。
- 如果两条路由来自相同的协议和 AS，则首选具有最小度量单位的路由。
- 将使用具有最小数字的下一跃点地址的路由。

分配优先级

对于 GateD 从其中接收路由的每个源，都将分配一个缺省优先级。优先级值的范围是 0 到 255，最小的数字表示最高优先级的路由。

下表总结以各种方式得到的路由的缺省优先级值。该表将列出设置优先级的语句（其中一些是语句中的子句），并显示每个语句适用的路由类型。此外还列出每种路由的缺省优先级，并且该表注明了协议之间的优先顺序。语句的范围越窄，所分配的优先级值的优先级就越高，但是它影响的路由集也越小。

优先级	定义语句	缺省值
直接连接网络	interface	0
OSPF 路由	ospf	10
IS-IS 1 级路由	isis level 1	15
IS-IS 2 级路由	isis level 2	18
内部生成的缺省值	gendefault	20
重定向	redirect	30
通过路由套接字得到的路由	kernel	40
来自 config 的静态路由	static	60
ANS SPF (SLSP) 路由	slsp	70
HELLO 路由	hello	90
RIP 路由	rip	100
点对点接口		110
关闭的接口的路由	interfaces	120

汇聚/生成路由	aggregate/generate	130
OSPF AS 外部路由	ospf	150
BGP 路由	bgp	170
EGP	egp	200

优先级规范示例

```
interfaces {
   interface 138.66.12.2 preference 10 ;
} ;
rip yes {
   preference 90 ;
} ;
import proto rip gateway 138.66.12.1 preference 75 ;
```

这些语句中，对于通过 RIP 从网关 138.66.12.1 得到的路由所应用的优先级是 75。适用于通过 RIP 从网关 128.66.12.1 得到的路由的优先级在 accept 语句中定义。适用于其他 RIP 路由的优先级可在 **rip** 语句中找到。在 interface 语句上设置的优先级仅适用于该接口的路由。

跟踪语句

跟踪语句控制跟踪选项。GateD 跟踪选项可以在多个级别上配置。跟踪选项包括文件规范、控制选项以及全局和协议特定的跟踪选项。除非被覆盖，否则下一个更高级别的跟踪选项将由较低的级别继承。例如，BGP 对等端跟踪选项从 BGP 组跟踪选项继承，BGP 组跟踪选项从全局 BGP 跟踪选项继承，全局 BGP 跟踪选项从全局 GateD 跟踪选项继承。在每一级别，跟踪规范均覆盖继承的选项。

全局跟踪选项

全局选项有两种：仅影响全局操作的选项和对协议具有潜在影响的选项。

仅具有全局影响

仅具有全局影响的跟踪标志包括：

parse 跟踪词汇分析程序和解析程序。主要由 GateD 开发人员用于调试。

adv 跟踪策略块的分配和释放。主要由 GateD 开发人员用于调试。

symbols 用于跟踪在启动时从内核读取的符号。指定该级别跟踪的唯一有用方法是通过命令行上的 **-t** 选项，这些因为符号是在分析配置文件之前从内核中读取的。

iflist 用于跟踪内核接口列表的读取。它可用于通过命令行上的 **-t** 选项指定该跟踪，这是因为第一次接口扫描在读取配置文件之前进行。

协议影响

对协议具有潜在影响的选项标志包括：

all 打开以下所有功能。

general 用于指定 **normal** 和 **route** 的简写表示法。

state 跟踪协议中的状态机转换。

normal 跟踪常规协议实例。始终会跟踪异常协议实例。

policy 跟踪协议和用户指定策略如何应用到导入和导出的路由。

task 跟踪系统接口和与该协议或对等端关联的处理。

timer 跟踪该协议或对等端的计时器使用情况。

route 跟踪该协议或对等端安装的路由的路由表更改。

并非以上所有选项均适用于所有协议。某些情况下，它们的使用毫无意义（例如 RIP 不具有状态机），某些情况下，请求的跟踪尚未实现（如 **policy** 选项的 RIP 支持）。

目前无法在命令行上指定数据包跟踪。这是因为数据包跟踪的全局选项可能会创建过多输出。

如果协议从全局跟踪选项继承其跟踪选项，则将隐藏无意义的跟踪级别（如 **parse**、**adv** 和数据包跟踪选项）。

全局跟踪语句具有直接的影响，尤其是在分析将影响配置文件分析的选项时。配置文件中指定的协议所继承的值最初从在分析时有效的全局选项继承（除非它们被更加特定的选项覆盖）。读取配置文件后，未显式指定的跟踪选项将从配置文件末尾的有效全局选项继承。

数据包跟踪

数据包跟踪非常灵活。对于任何给定的协议，存在一个或多个数据包跟踪选项。所有协议均允许使用 **packets** 关键字来跟踪该协议发送和接收的所有数据包。大多数协议均具有其他选项来将跟踪限制到有用的数据包类型子集。这些跟踪选项可以使用以下修饰符进行进一步控制：

detail **detail** 必须在 **send** 或 **recv** 之前指定。通常，以一行或两行的简洁形式跟踪数据包。指定 **detail** 时，将使用更加详细的格式来提供有关数据包内容的详细信息。

send
recv 这些选项将跟踪限制到发送或接收的数据包。如果没有这些选项，将跟踪发送和接收的数据包。

如果指定了 **Detail**，则其必须位于 **send** 或 **recv** 之前。如果协议允许多种不同类型的数据包跟踪，则修饰符可以应用于各个单独的类型。但是要注意，在一个跟踪规范中将对跟踪标记进行汇总，因此指定 **detail packets** 将打开所有数据包的完全跟踪功能。

跟踪选项语法

traceoptions ["*trace_file*" [**replace**] [**size** *size*[**k**|**m**] **files** *files*]]
[*control_options*] *trace_options* [**except** *trace_options*] ;

traceoptions none ;

trace_file 指定用于接收跟踪信息的文件。如果该文件名不以斜线 (**/**) 开头，则将启动 GateD 所在的目录以将添加到该名称之前。

replace 跟踪应该以替换现有文件开始。缺省操作是追加到现有文件。

size *size*[**k**|**m**] **files** *files*

将跟踪文件的最大大小限制到指定的大小（最小 10K）。跟踪文件达到指定大小时，将重命名为 **file.0**、**file.1**、 **file.2**，直到最大的文件数（最小指定值是 2）。

control_options

指定用于控制跟踪形式的选项。有效值包括：

nostamp

指定时间戳不应添加到所有跟踪行之前。

except *trace_options*

用于启用范围较宽的跟踪类，然后禁用较为特定的选项。

none 指定应该关闭该协议或对等端的所有跟踪功能。

指令语句

指令语句向 GateD 配置语言分析程序提供有关包含文件的指示以及这些文件所在的目录。指令语句由分析程序立即执行。其他语句以分号 (**;**) 终止，但是指令语句以换行符终止。两个指令语句为：

%directory "*directory*"

定义存储包含文件的目录。如果使用，则 GateD 将在路径名标识的目录中查找任何不具有完全限定路径名的包含文件，例如不以“/”开头的文件。该语句不会实际更改当前目录，而仅指定应用于包含文件名的前缀。

%include "*filename*"

标识包含文件。该文件的内容包含在 gated.conf 文件中出现 **%include** 指令的位置。如果文件名未完全限定（不以“/”开头），则将其视为相对于在 **%directory** 指令中定义的目录。通过 **%include** 指令语句，将先完全分析指定的文件，然后再继续处理该文件。支持最多十个级别的嵌套。最大嵌套级别可以通过更改 **FI_MAX** 在 **parse.h** 中的定义来增加。

在复杂的环境中，将较大的配置分为多个更易于理解的小段可能会很有帮助，不过 GateD 最大的优点之一就是将包含多个不同路由协议的配置组合到单个文件中。将小文件分段会使路由配置变得不必要的复杂。

选项语句

选项语句允许指定某些全局选项。如果使用，则 **options** 必须出现在 gated.conf 文件中其他任何类型配置语句之前。

选项语句语法为：

options
  [**nosend**]
  [**noresolv**]

[**gendefault** [**preference** *preference*] [**gateway** *gateway*]]
[**syslog** [**upto**] *log_level*]
[**mark** *time*]
;

选项列表可以包含一个或多个以下选项：

gendefault [**preference** *preference*] [**gateway** *gateway*]

如果启用了 **gendefault**，当 BGP 或 EGP 邻居运行时，将导致以特殊协议 **default** 创建缺省路由。这可以通过 **nogendefault** 选项按 BGP/EGP 组禁用。缺省情况下，该路由的优先级是 20。该路由通常不安装在内核转发中，其存在目的只是为了向其他协议声明。如果指定了网关，则缺省路由将以所列网关的下一跃点安装在内核转发表中。

请注意，使用更普通的选项优于使用该 **gendefault** 选项。**gendefault** 选项可能会在将来的版本中不再使用。有关 **generate** 语句的详细信息，请参阅“路由汇聚”一节。

nosend 不发送任何数据包。通过该选项，可以在活动网络上运行 GateD 来测试协议交互，而不用实际参与路由协议。可以检查 GateD 日志中的数据包跟踪，来验证 GateD 是否运行正常。它对于 RIP 和 HELLO（且可能包括 SMUX SNMP 接口）最为有用。该选项还不适用于 BGP，并且对于 EGP 和 OSPF 不太有用。

noresolv

缺省情况下，GateD 将尝试使用 **gethostbyname()** 和 **getnetbyname()** 库调用将符号名称解析为 IP 地址。这些调用通常使用域名系统 (DNS)，而不是 hosts 本地主机和网络表。如果路由信息不足，无法发送 DNS 查询，则 GateD 将在启动期间出现死锁。该选项可用于禁止这些调用，因为符号名称将导致配置文件错误。

syslog [**upto**] *log_level*

通过 syslog 控制支持 **setlogmask()** 的系统上的数据 GateD 日志数量。可用日志记录级别和其他术语与 *setlogmask*(3) 联机帮助页中的定义相同。缺省值等效于 **syslog upto info**。

mark *time*

如果指定该选项，则 GateD 可以指定的间隔时间将消息输出到跟踪日志。它可以用作一种确定 GateD 是否仍在运行的方法。

接口语句

接口语法

interfaces {
options
[**strictinterfaces**]
[**scaninterval** *time*]
[**aliases-nh** (**primary** | **lowestip** | **keepall**)]
;
interface *interface_list*

[**preference** *preference*]
[**down preference** *preference*]
[**passive**]
[**simplex**]
[**reject**]
[**blackhole**]
[**alias primary** *address*]
[**aliases-nh** (**primary** | **lowestip** | **keepall**)]
;
define *address*
[**broadcast** *address*] | [**pointtopoint** *address*]
[**netmask** *mask*]
[**multicast**]
;
} ;

接口是路由器及其连接网络之间的连接。物理接口可以按接口名、 IP 地址或域名指定（除非网络是未编号的点对点网络）。通过配置语言中的多个引用级别，可以使用通配符、接口类型名或删除单词地址来标识接口。由于将来的 Unix 操作系统可能允许每个接口使用多个地址，因此应使用接口名时应谨慎。interface_list 是一个或多个接口名的列表，其中包括通配符名称（不包含数字的名称）以及可指定多个接口、地址或所有接口的标记 **all** 的名称。

options

允许配置某些与接口相关的全局选项。其中包括：

strictinterfaces

表示在 GateD 启动且未在 **define** 语句中列出时，如果在配置文件中引用不存在的接口，则将导致致命错误。如果不使用该选项，将发出警告消息，但 GateD 将继续操作。

scaninterval *time*

指定 GateD 扫描内核接口列表以进行更改的频率。大多数系统的缺省值是每 15 秒，在通过路由套接字 (BSD 4.4) 传递接口状态更改的系统上则是 60 秒。请注意，GateD 还将在收到 **SIGUSR2** 时扫描接口列表。

aliases-nh (**primary** | **lowestip** | **keepall**)

指定 GateD 在向类似 Serviceguard 环境的接口分配的多个地址时，将安装作为接口路由的下一跃点的地址。如果使用 **primary** ，则将安装主接口地址（缺省）。如果使用 **lowestip** ，则将安装具有最小 IP 地址的地址。如果使用 **keepall** ，所有接口路由将保留在内核中，最多可保留 **RT_N_MULTIPATH** 条路由。这是编译时常量。它是可使用接口选项为接口覆盖的全局参数。

注释：在 Serviceguard 环境中使用 GateD 时， **keepall** 选项是必需的。

interface *interface_list*

设置指定接口上的接口选项。接口列表是 **all** 或接口名（请参阅有关接口名的“警告”）、域名或数字地址的列表。该语句的可用选项包括：

preference *preference*

在该接口启动且看起来处于正常运行状态时，设置其路由的优先级。缺省值是 **0**。

down preference *preference*

在 GateD 认为该接口不能正常运行，但内核未指示它已关闭时设置其路由的优先级。缺省值是 **120**。

passive 在由于收到的路由信息不足而认为该接口运行不正常时，防止 GateD 更改其路由的优先级。只有在接口积极参与路由协议时， GateD 才将执行该检查。

simplex

将接口定义为无法监听其自己的广播数据包。某些系统使用 **IFF_SIMPLEX** 标志将接口定义为单工，在其他系统上，需要在配置文件中指定。在单工接口上，将假定来自用户自己的数据包已经在软件中送回，而不将其用作接口正常运行的指示信号。

reject 指定在内核中安装 *reject* 路由时，匹配这些条件的接口的地址将用作本地地址。只应该用于已安装了 reject/blackhole 伪接口的基于 BSD 4.3 Tahoe 或更早版本的系统。

blackhole

指定在内核中安装 *reject* 路由时，匹配这些条件的接口的地址将用作本地地址。只应该用于已安装了 reject/blackhole 伪接口的基于 BSD 4.3 Tahoe 或更早版本的系统。

alias primary *address*

指定该接口的主地址 *address* 。该选项将覆盖 GateD 确定为主地址的地址。

aliases-nh (**primary** | **lowestip** | **keepall**)

指定 GateD 在向类似 Serviceguard 环境的接口分配多个地址时，将安装作为与该接口关联的路由的下一跃点的地址。如果使用 **primary** ，则将安装主接口地址（缺省）。如果使用 **lowestip** ，则将安装具有最低 IP 地址的地址。如果使用 **keepall** ，则所有接口路由都将保留在内核中，最多可保留 **RT_N_MULTIPATH** 条路由。这是编译时常量。该参数覆盖当前接口的全局选项。

注释：在 Serviceguard 环境中使用 GateD 时， **keepall** 选项是必需的。

define *address*

定义在 GateD 启动时可能不存在的接口，以便在定义 **strictinterfaces** 时可以在配置文件中引用它们。可能的 **define** 关键字包括：

broadcast *address*

将接口定义为支持广播（以太网或令牌环），并指定广播地址。

pointtopoint *address*

将接口指定为点对点接口（SLIP 或 PPP），并指定本地端的地址。 **define** 语句上的第一个地址引用接口 **remote** 端的主机地址，该 **pointtopoint** 关键字之后指定的地址定义接口 **local** 端的地址。

假定未定义为广播或点对点的接口是非广播多路访问 (NBMA)，如 X.25 网络。

netmask *mask*

指定将在该接口上使用的子网掩码。在点对点接口上会将其忽略。

multicast

指定该接口支持多播。

接口列表

接口列表是对接口或接口组的引用的列表。可以通过四种方法来引用接口。下面按从最普遍到最具体的顺序列出这些方法。

all 它引用所有可用的接口。

Interface name wildcard

它引用相同类型的所有接口。Unix 接口由设备驱动程序的名称（如 **ie**）和一个单位数字（如 **0**、**5** 或 **22**）组成。名称引用仅包含字母字符，并且匹配任何包含相同字母部分的接口。

例如，Sun 上的 **ie** 引用所有 Interlan Ethernet 接口，**le** 引用所有 Lance Ethernet 接口。但是 **ie** 不匹配 **iel0**。

Interface name

它引用特定的接口，通常是一个物理接口。它们的指定格式是一个字母部分后接一个数字部分。它将匹配一个特定的接口。但是请注意，在许多系统上，一个给定的物理接口可能存在多个协议 (*IP*) 地址。例如，**ef1** 将匹配名为 **ef1** 的接口，但不匹配名为 **ef10** 的接口。

Interface address

它匹配一个特定的接口。可以通过协议地址 (*10.0.0.51*) 或符号主机名 (*nic.ddn.mil*) 进行引用。请注意，符号主机名引用只有在仅解析为一个地址时才有效。不建议使用符号主机名。

如果配置文件中存在多个接口列表，它们包含多个参数，则将在运行时收集这些参数，为给定的接口创建特定的参数列表。如果在多个列表上指定同一参数，则将使用与最具体的接口对应的参数。

例如，考虑包含三个接口（**le0**、**le1** 和 **du0**）的系统。

```
rip yes {
   interface all noripin noripout ;
   interface le ripin ;
   interface le1 ripout ;
} ;
```

将仅接受来自接口 **le0** 和 **le1**（但不是 **du0**）的 RIP 数据包。RIP 数据包仅在接口 **le1** 上发送。

IP 接口地址和路由

BSD 4.3 和更高版本的联网实现允许四种类型的接口。某些实现允许每个物理接口使用多个协议地址，它们大多基于 *BSD 4.3 Reno* 或更高版本。

loopback

该接口必须具有地址 **127.0.0.1**。发送到该接口的数据包将送回给初始发送方。该接口也用作综合接口，用于实现其他功能，如 *reject* 和 *blackhole* 路由。虽然该接口上会报告网络掩码，但会将其忽略。一种比较有用的做法是给该接口分配一个与 OSPF 或 BGP 路由器 *ID* 相同的附加地址；这样，就可以基于路由器 *ID* 路由到在某些接口关闭时仍将工作的系统。

broadcast

它是支持物理级广播（如以太网、令牌环和 *FDDI*）的多路访问接口。该接口具有关联的子网掩码和广播地址。广播网络的接口路由将是整个子网的路由。

point-to-point

它是到其他主机的通道，通常位于某种串行链接上。该接口具有一个本地地址和一个远程地址。虽然可以为一个点对点接口指定多个地址，但是一般没有合理的理由来这样做。

远程地址必须在给定路由器上的所有接口地址之中是唯一的。本地地址可以在多个点对点和最多一个非点对点接口之间共享。这从技术上看是无地址链接的一种路由器 *ID* 方法。使用该技术时，如果不需要任何子网，则将保存子网。

如果在点对点接口上指定子网掩码，则它仅由 RIP v1 和 HELLO 用来确定可以传播到该接口另一端的路由器的子网。

nonbroadcast multiaccess or **nbma**

该类型的接口是多路访问的，但是不支持广播。其示例包括帧中继和 *X.25*。该类型的接口具有一个本地地址和一个子网掩码。

GateD 确保每个已配置并运行的 IP 接口均有一条可用的路由。通常，这由配置接口的 *ifconfig* 命令来完成；GateD 执行该操作来确保一致性。

对于点对点接口，GateD 将安装一些专用路由。如果一个或多个点对点接口上的本地接口没有与非点对点接口共享，则 GateD 将安装连接到本地地址的路由，该路由指向优先级为 110 的环回接口。这将确保在本地处理源自以该本地地址为目标的主机的数据包。 OSPF 首选通过点对点链接路由本地接口的数据包，在该链接中，远程端的路由器将返回这些数据包。它用于验证链接的操作。由于 OSPF 安装了优先级为 10 的路由，因此这些路由将覆盖安装的优先级为 110 的路由。

如果一个或多个点对点接口的本地地址与一个非点对点接口共享， GateD 将安装优先级为 0 的本地接口的路由，该路由将不会在转发表中安装。在当前系统将用作主机时，这可以防止 OSPF 等协议通过串行接口将数据包路由到该地址。

当接口的状态更改时，GateD 将通知执行适当的操作。GateD 假定不存在未标记为 *UP* 的接口。虽然这可能不是最正确的操作，但目前的工作方式就是如此。

GateD 忽略任何包含对于本地、远程或广播地址或者子网掩码无效的数据的接口。无效数据在任何字段中都包含

零。 GateD 还将忽略任何具有相同本地和远程地址的点对点接口，它假定该接口处于某种环回测试模式。

定义语句

定义语句是与所有 GateD 或者至少多个协议相关的通用配置语句。三个定义语句为 **autonomoussystem**、**routerid** 和 **martians**。如果使用， **autonomoussystem**、**routerid** 和 **martians** 必须出现在 gated.conf 文件中其他类型的配置语句之前。

自治系统配置

autonomoussystem *autonomous_system* [**loops** *number*] **;**

将该路由器的自治系统编号设置为 **autonomous system**。如果正在使用 BGP 或 EGP，则必须使用该选项。AS 编号由网络信息中心 (NIC) 分配。

Loops 仅用于支持 AS 路径的协议，如 BGP。它控制该自治系统可以在 AS 路径中出现的次数，并且缺省为 1（一次）。

路由器 **ID** 配置

routerid *host* **;**

设置路由器标识符，以供 BGP 和 OSPF 协议使用。缺省值是 GateD 遇到的第一个接口的地址。非点对点接口的地址比点对点接口的本地地址具有更高的优先级，具有最高优先级的是环回接口上不是环回地址 (127.0.0.1) 的地址。

martians 配置

martians {
  host *host* [**allow**] **;**
    network [**allow**] **;**
    network **mask** *mask* [**allow**] **;**
    network **masklen** *number* [**allow**] **;**
  default [**allow**] **;**
} ;

定义一个忽略所有路由信息的 *martians* 地址的列表。错误配置的系统有时会发出明显无效的目标地址。这些无效地址称作 martians，路由软件将拒绝这些地址。通过该命令，可以在 martians 地址列表中进行添加。有关指定范围的详细信息，请参阅“路由过滤”一节。此外，可以指定 **allow** 参数，以显式允许被禁用的范围子集。

示例定义语句

```
options gendefault ;
autonomoussystem 249 ;
interface 128.66.12.2 passive ;
martians {
  0.0.0.26
  };
```

示例中的语句执行以下功能:

- options 语句指示系统在它与 EGP 或 BGP 邻居建立对等关系时生成缺省路由。
- autonomoussystem 语句指示 GateD 在 EGP 和 BGP 中使用 249 作为 AS 编号。
- interface 语句指示 GateD 不要将接口 128.66.12.2 标记为关闭（如果未检测到通信量）。
- martians 语句防止接受到 0.0.0.26 的路由。

协议概述

路由协议确定到达每个目标的“最佳”路由，并在网络上的系统之间分发路由信息。路由协议分为两个通用组：内部协议和外部协议。 GateD 软件在一个软件守护程序中合并了内部和外部路由协议的管理。

内部路由协议

内部协议用于自治系统 (AS) 内交换连接性信息。它们是以缩写 **igp** 表示的一个类。下面是几种内部协议：

RIP RIP（Routing Information Protocol，路由信息协议）版本 1 和版本 2 是最常用的内部协议。RIP 选择具有最小度量单位的路由作为最佳路由。该度量单位是指跃点数，表示数据到达其目标之前必须通过的网关数。RIP 可接受的最长路径是 15 个跃点。如果度量单位大于 15，则认为目标无法连接，GateD 将忽略该路由。RIP 假定最佳路由是使用最少网关的路由，它是最短的路径，因为不用考虑路由上的堵塞或延迟。

RIP v1 协议在 RFC 1058 中描述，RIP v2 协议在 RFC 1388 中描述。

HELLO

另一个内部协议 HELLO 在选择最佳路由时使用中继作为决定因素。往返时间是数据报从源和目标传递所用的时间。HELLO 对于 Internet 具有历史意义，它是在最初的原型 NSFNET 主干 *fuzzball* 网关之间使用的协议。如今，HELLO 与 fuzzball 一样是很少使用的协议。

HELLO 协议的较早版本在 RFC 891 中进行了描述。

OSPF

OSPF（Open Shortest Path First，开放式最短路径优先）是链接状态协议。OSPF 比 RIP 更适合于包含多个路由器的复杂网络。 OSPF 提供同成本多路径路由。

OSPF 在 RFC 1583 中进行了描述， MIB 在 RFC 1253 中定义。其他相关文档包括 RFC 1245、 RFC 1246 和 RFC 1370。

外部路由协议

外部协议用于在自治系统之间交换路由信息。只有当一个自治系统必须与其他自治系统交换路由信息时，外部协议才是必需的。自治系统中的路由器运行内部路由协议，如 RIP。只有将一个自治系统连接到另一个自治系统的网关才需要运行外部路由协议。 GateD 当前支持两种外部协议：

EGP

EGP（Exterior Gateway Protocol，外部网关协议）： EGP 连接性信息最初传递到 ARPANET/MILNET“核心”网关中，在其中将选择最佳路由，并将这些路由传回所有连接的自治系统。随着 Internet 趋向于较低层次化的体系结构，采用层次化结构的外部路由协议 EGP 的有效性也逐渐降低。

EGP 协议在 RFC 827 和 RFC 904 中有所说明。

BGP

BGP（Border Gateway Protocol，边界网关协议）正在替代 EGP 成为外部协议的最佳选择。BGP 在自治系统之间交换可达性信息，但却提供比 EGP 更多的功能。BGP 使用路径属性来提供有关每条路由的详细信息，以协助选择最佳的路由。例如，路径属性可能包括基于路由决策中涉及的政治、组织或安全（策略）考虑事项的管理首选项。 BGP 支持非层次化拓扑结构，可用于实现等效自治系统的网络结构。

BGP 版本 1 在 RFC 1105 中进行了描述，版本 2 在 RFC 1163 中进行了描述，版本 3 在 RFC 1267 中进行了描述。版本 3 MIB 在 RFC 1269 中进行了描述。 RFC 1164 和 RFC 1268 两个文档介绍版本 2 和 3 在 Internet 中的应用。 RFC 1265 和 RFC 1266 中提供了 BGP 版本 3 的协议分析和实用经验。 RFC 1397 介绍如何在 BGP 版本 2 和 3 中公告缺省路由。最后， RFC 1403 介绍了 BGP - OSPF 交互。

其他路由协议

Router Discovery

路由器发现协议用于向主机通知它可向其发送数据包的主机是否可用，并可用于补充静态配置的缺省路由器。这是运行主机的首选协议，将阻止这些主机窃听路由协议。

路由器发现在 RFC 1256 中进行了描述。

路由信息协议 **(RIP)**

使用最广泛的内部网关协议之一是路由信息协议 (RIP)。RIP 是距离矢量（或用于本地网络的 Bellman-Ford 路由协议）的实现。它将路由器分类为主动路由器和被动路由器（无提示）。主动路由器向其他路由器公告其路由（连接性信息）；被动路由器则根据公告监听并更新其路由，但不进行公告。通常，路由器在主动模式下运行 RIP，而主机则在被动模式下运行 RIP。

在主动模式下运行 RIP 的路由器按照设定的时间间隔广播更新。每一更新包含配对值，其中每一对值由 IP 网络地址和该网络的整数距离组成。 RIP 使用跃点数度量单位来测量到目标的距离。在 RIP 度量单位中，路由器公告直接连接的网络的度量单位为 1。可通过另一个网关到达的网络的跃点数是 2，依此类推。因此，从一个给定源到一个给定目标的路径上的跃点数或跃点计数是指数据报将在该路径上遇到的的网关数。使用跃点数来计算最短路径并不总能得到最佳结果。例如，跃点数为 3 且跨越三个以太网的路径可能比跃点数为 2 且跨越两个慢速串行线路的路径快得多。为了弥补技术上的不足，许多路由器为慢速链接公告虚假的跃点数。

RIP 随大多数 UNIX 系统提供，它由路由守护程序 **routed**（英语发音为 route-"d"）运行。RIP 路由守护程序将动态地累积通过 RIP 更新收到的信息。启动时，它将发出对路由信息的请求，然后监听对该请求的响应。如果配置为支持 RIP 的系统监听到请求，则它将根据其路由数据库中的信息用 RESPONSE 数据包做出响应。RESPONSE 数据包包含目标网络地址和每个目标的路由度量单位。

收到 RIP RESPONSE 数据包时，路由守护程序将用其中的信息重建路由数据库，方法是将新路由和“更佳”（较小度量单位）的路由添加到该数据库中已列出的目标中。如果该目标的下一个路由器声明路由包含超过 15 个跃点，或者路由已删除，则 RIP 也将从数据库中删除路由。如果在指定的时间内没有从该网关收到更新，则将删除通过该网关的所有路由。通常，路由更新会每 30 秒钟发送一次。在许多实现中，如果在 180 秒内没有收到某网关的响应，则将从路由数据库删除该网关中的所有路由。这一 180 秒间隔时间也适用于删除特定路由。

RIP 版本 2（通常称作 RIP II）为 RIP 添加了附加功能。其中某些功能与 RIP I 兼容，有些功能则不兼容。要避免向 RIP I 路由提供可能会被误解的信息，当多播其数据包时，RIP II 只能使用不兼容的功能。在不支持 IP 多播的接口上，将使用不包含可能导致误解信息的与 RIP I 兼容的数据包。

RIP II 的一些最显著的改进包括：

下一跃点

主要的功能是公告下一跃点，以使用路由器之外的设备提供路由更新。向不运行 RIP 的静态路由器公告静态路由时，可避免通过静态路由器指定数据包时必须两次跨越网络，因此它非常有用。

RIP I 路由器将忽略 RIP II 数据包中的下一跃点信息。这可能导致数据包两次跨越网络（与 RIP I 的情况完全相同）。因此该信息在兼容 RIP I 的 RIP II 数据包中提供。

网络掩码

RIP I 假定给定网络的所有子网均具有相同的网络掩码。它利用该假定来计算收到的所有路由的网络掩码。该假定将防止在 RIP 数据包中包含具有不同网络掩码的子网。RIP II 添加了该功能，以在数据包中显式指定每个网络的网络掩码。

RIP I 路由器将忽略 RIP II 数据包中的网络掩码，因为其网络掩码的计算很可能是错误的。出于这个原因，兼容 RIP I 的 RIP II 数据包决不能包含可能被误解的网络。这些网络必须仅在作为多播的本机 RIP II 数据包中提供。

身份验证

RIP II 数据包还可以包含两种身份验证字符串之一，它们可用于验证所提供的路由数据的有效性。身份验证可以在兼容 RIP I 的 RIP II 数据包中使用，但请注意，RIP I 路由器会将其忽略。

第一种方法是简单的口令，在该方法中，数据包内包含最多 16 个字符的身份验证密钥。如果它不匹配预期的密钥，则将忽略数据包。由于可以通过观察 RIP 数据包来得到身份验证密钥，因此该方法提供了非常低的安全性。

第二种方法仍处在试验阶段，可能在将来的版本中以不兼容的方式进行更改。该方法使用 MD5 算法创建 RIP 数据包的加密校验和以及最多 16 个字符的身份验证密钥。传输的数据包不包含身份验证密钥本身，而包含加密校验和，称作 *digest*。接收方路由器将使用正确的身份验证密钥执行计算，如果 digest 不匹配，则忽略数据包。此外，将维护一个序列号，以防止重放较旧的数据包。通过该方法，可以在很大程度上确保源自路由器的路由数据具有有效的身份验证密钥。

每个接口可以指定两种身份验证方法。数据包始终使用主方法发送，但收到的数据包将在忽略之前使用主方法和备用方法进行检查。此外，独立的身份验证密钥将用于非路由器查询。

RIP-I 和网络掩码

RIP-I 从通过其收到数据包的接口的网络掩码派生收到的网络和主机的网络掩码。如果收到的网络或主机与通过其进行接收的接口位于相同的自然网络上，并且将网络划分为子网（指定掩码比自然网络掩码更为具体），则将子网掩码用于目标。如果设置了掩码外的位，则假定其为主机。否则，假定为子网。

在点对点接口上，网络掩码用于远程地址。如果网络掩码匹配远程地址的自然网络或者都是一，则将忽略这些接口上的网络掩码。

与以前的版本不同，将忽略零子网掩码（匹配接口的自然网络，但具有更具体或更长的网络掩码的网络）。如果不需要该掩码，则可以使用路由过滤器将其拒绝。

RIP 语句

rip yes | **no** | **on** | **off** [**{**
 broadcast ;
 nobroadcast ;
 nocheckzero ;
 preference *preference* **;**
 defaultmetric *metric* **;**
 query authentication [**none** | [[**simple**|**md5**] *password*]] **;**
 interface *interface_list*
  [**noripin**] | [**ripin**]
  [**noripout**] | [**ripout**]
  [**metricin** *metric*]
  [**metricout** *metric*]
  [**version 1**]|[**version 2** [**multicast**|**broadcast**]]
  [[**secondary**] **authentication** [**none** | [[**simple**|**md5**] *password*]] **;**
 trustedgateways *gateway_list* **;**
 sourcegateways *gateway_list* **;**
 traceoptions *trace_options* **;**
}] **;**

rip 语句可启用或禁用 RIP。如果未指定 **rip** 语句，则缺省值为 **rip on ;** 。如果启用，RIP 将在仅存在一个接口时假定 **nobroadcast**，在存在多个接口时假定 **broadcast**。

其选项如下所示：

broadcast
指定无论存在多少个接口，仍将广播 RIP 数据包。在将静态路由或从其他协议得到的路由传播到 RIP 时，该选项非常有用。某些情况下，如果仅存在一个网络接口，则使用 **broadcast** 可能导致数据包两次遍历单个网络。

nobroadcast
指定 RIP 数据包将不在连接的接口上广播（即使存在多个接口）。如果存在 **sourcegateways** 子句，则路由仍将直接单播到该网关。

nocheckzero
指定 RIP 不应确保传入的版本 1 RIP 数据包中的保留字段为零。RIP 通常会拒绝其中的保留字段为零的数据包。

preference *preference*

设置从 RIP 得到的路由的优先级。缺省优先级是 100。该优先级可以用导入策略中指定的优先级覆盖。

defaultmetric *metric*

定义在通过 RIP 声明从其他协议得到的路由时使用的度量单位。如果未指定，缺省值为 16（无法连接）。这种值选择要求您显式指定度量单位，以将路由从其他协议导入至 RIP 中。该度量单位可以用导出策略中指定的度量单位覆盖。

query authentication [**none** | [[**simple**|**md5**] *password*]] **;**

指定不源自路由器的查询数据包所必需的身份验证。缺省值为 *none*。

interface *interface_list*

控制在特定接口上发送 RIP 的各种属性。有关 *interface_list* 的说明，请参阅有关接口列表规范的部分。

请注意，如果同一个子网上配置了多个接口， RIP 更新将只从配置 RIP 输出的第一个接口发送。该限制是由 Unix 内核的运行方式决定的。希望在将来的版本中将其删除。

可能的参数包括:

noripin

指定将忽略通过指定接口接收的 RIP 数据包。缺省为监听所有非环回接口上的 RIP 数据包。

ripin 这是缺省值。在通配符接口描述符上使用 **noripin** 时，该参数是必需的。

noripout

指定将不在指定的接口上发送 RIP 数据包。缺省操作是在 **broadcast** 模式下在所有广播和非广播接口上发送 RIP。必须手动配置在点对点接口上发送 RIP 的方式。

ripout 这是缺省值。当需要在点对点接口上发送 RIP 时，该参数是必需的；在通配符接口描述符上使用 **noripin** 时，该参数可能是必需的。

metricin *metric*

指定在路由表中安装传入路由之前，向其添加的 RIP 度量单位。缺省值是内核接口度量单位加 1（即缺省的 RIP 跃点数）。如果指定该值，则将其用作绝对值。将不添加内核度量单位。该选项用于使该路由器首选从其他接口得到的 RIP 路由（相比通过指定接口得到的 RIP 路由）。

metricout *metric*

指定要添加到通过指定接口发送的路由的 RIP 度量单位。缺省值为零。该选项用于使其他路由器首选优先于该路由器的 RIP 路由的其他来源。

version 1

指定通过指定接口发送的 RIP 数据包将是版本 1 数据包。这是缺省值。

version 2

指定将在指定的接口上发送 RIP 版本 2 数据包。如果在该接口上可用 IP 多播支持，则缺省值是发送完整的版本 2 数据包。如果不可用，将发送兼容版本 1 的版本 2 数据包。

multicast

指定应该在该接口上多播 RIP 版本 2 数据包。这是缺省值。

broadcast

指定即使可以使用 IP 多播，仍应该在该接口上广播兼容 RIP 版本 1 的版本 2 数据包。

[**secondary**] **authentication** [**none** | [**simple**|**md5**] *password*]

它定义要使用的身份验证类型。它仅适用于 RIP 版本 2，对于 RIP-1 数据包将忽略。缺省的身份验证类型是 **none**。如果指定了口令，则身份验证类型缺省为 **simple**。口令应该是包含 0 到 16 个字符的引用字符串。

如果指定了 **secondary**，则定义二级身份验证。如果省略，则指定主身份验证。缺省值是主要身份验证 *none* 以及无二级身份验证。

trustedgateways *gateway_list*

定义 RIP 将接受其中的更新的网关列表。 *gateway_list* 只是主机名或 IP 地址的列表。缺省情况下，共享网络上的所有路由器都受到信任，可提供路由信息。但是如果指定了 **trustedgateways** 子句，则仅接受来自列表中网关的更新。

sourcegateways *gateway_list*

定义 RIP 向其直接（而不是通过多播或广播）发送数据包的路由器列表。它可用于向特定网关发送不同的路由信息。对该列表中网关的更新不受接口上 **noripout** 的影响。

traceoptions *trace_options*

指定 RIP 的跟踪选项。（请参阅下面的跟踪语句和 RIP 特定跟踪选项）。

跟踪选项

每当声明新路由，所声明的度量单位更改或者路由进入或离开锁禁状态时，**policy** 选项将记录相应信息。

数据包跟踪选项（可以使用 **detail**、**send** 或 **recv** 修改）：

packets 所有 RIP 数据包。

request RIP 信息请求数据包，如 **REQUEST**、**POLL** 和 **POLLENTRY**

response RIP **RESPONSE** 数据包，其类型是实际包含路由信息的数据包的类型。

other 任何其他类型的数据包。唯一有效的数据包是 **TRACE_ON** 和 **TRACE_OFF**，它们均被忽略。

Hello 协议

除非有特定的需要，否则最好不要使用 HELLO。我们计划在 GateD 4.0 左右将其删除。

HELLO 协议是一种内部协议，它根据在源和目标之间发送数据包所用的时间来使用路由度量单位。 HELLO 数据包带有时间戳信息，利用该信息，接收方可以计算到目标的最短延迟路径。“最佳”路由是延迟时间最短的路由。HELLO 中使用的时间单位是毫秒。如果 HELLO 更新数据包使用少于 100 毫秒的时间在两个路由器之间传输，则将最小值 100 用于该跃点。因此，在由高速接口组成的网络上，HELLO 实际上缺省为使用跃点数。与任何路由算法一样，HELLO 不能太快地更改路由，否则它将变得不稳定。要避免不稳定的情况， HELLO 的实现内置了滞后机制，直到确定更改将持久时才会更改路由。

缺省情况下，HELLO 与 RIP 类似，使用通过 **ifconfig** 命令设置的内核接口度量单位在将路由安装到路由表 (**metricin**) 时影响添加到路由的度量单位。由于内核接口度量单位是以跃点计量，因此必须将其转换为 HELLO 毫秒度量单位。为此，将使用下表：

跃点	HELLO 计量
0	0
1	100
2	148
3	219
4	325
5	481
6	713
7	1057
8	1567
9	2322
10	3440
11	5097
12	7552
13	11190
14	16579
15	24564
16	30000

HELLO 和网络掩码

HELLO 从通过其收到数据包的接口的网络掩码派生收到的网络和主机的网络掩码。如果收到的网络或主机与通过其进行接收的接口位于相同的自然网络上，并且将网络划分为子网（指定掩码比自然网络掩码更为具体），则将子网掩码用于目标网络。如果设置了掩码外的位，则假定其为主机。否则，假定为子网。

在点对点接口上，将向远程地址提供网络掩码。如果网络掩码匹配远程地址的自然网络或者都是一，则将忽略这些接口上的网络掩码。

与以前的版本不同，将忽略零子网掩码（匹配接口的自然网络，但具有更具体或更长的网络掩码的网络）。如果不需要该掩码，则可以使用路由过滤器将其拒绝。

Hello 语句

hello yes | **no** | **on** | **off** [{
  broadcast ;
  nobroadcast ;
  preference *preference* **;**
  defaultmetric *metric* **;**
  interface *interface_list*
    [**nohelloin**] | [**helloin**]
    [**nohelloout**] | [**helloout**]
    [**metricin** *metric*]
    [**metricout** *metric*] **;**
  trustedgateways *gateway_list* **;**
  sourcegateways *gateway_list* **;**
  traceoptions *trace_options* **;**
}] **;**

hello 语句可启用或禁用 HELLO。如果未指定 **hello** 语句，则缺省值为 **hello off**。如果启用，HELLO 将在仅存在一个接口时假定 **nobroadcast**，在存在多个接口时假定 **broadcast**。

broadcast
指定无论存在多少个接口，仍将广播 HELLO 数据包。在将静态路由或从其他协议得到的路由传播到 HELLO 时，该选项非常有用。某些情况下，如果仅存在一个网络接口，则使用 **broadcast** 可能导致数据包两次遍历单个网络。

nobroadcast
指定 HELLO 数据包将不在连接的接口上广播（即使存在多个接口）。如果存在 **sourcegateways** 子句，则路由仍将直接单播到该网关。

preference *preference*
设置从 HELLO 得到的路由的优先级。缺省优先级是 90。该优先级可以用导入策略中指定的优先级覆盖。

defaultmetric *metric*
定义在通过 HELLO 公告从其他协议得到的路由时使用的度量单位。如果未指定，则缺省值为 30000（无法连接）。这种值选择要求您显式指定度量单位，以将路由从其他协议导入至 HELLO。该度量单位可以用导出策略中指定的度量单位覆盖。

interface *interface_list*
控制在特定接口上发送 HELLO 的各种属性。有关 *interface_list* 的说明，请参阅有关接口列表规范的部分。

请注意，如果同一个子网上配置了多个接口，则 HELLO 更新将只从配置 HELLO 输出的第一个接口发送。该限制是由 Unix 内核的运行方式决定的。希望在将来的版本中将其删除。

可能的参数包括：

nohelloin
指定将忽略通过指定接口接收的 HELLO 数据包。缺省为监听所有非环回接口上的 HELLO。

helloin 这是缺省值。在通配符接口描述符上使用 **nohelloin** 时，该参数是必需的。

nohelloout
指定将不在指定的接口上发送 HELLO 数据包。缺省操作是在 **broadcast** 模式下在所有广播和非广播接口上发送 HELLO。必须手动配置在点对点接口上发送 HELLO 的方式。

helloout
这是缺省值。当需要在点对点接口上发送 HELLO 时，该参数是必需的；在通配符接口描述符上使用 **nohelloin** 时，该参数可能是必需的。

metricin *metric*
指定在路由表中安装传入路由之前，向其添加的 HELLO 度量单位。缺省值是内核接口度量单位加 1（即缺省的 HELLO 跃点数）。如果指定该值，则将其用作绝对值。将不添加内核度量单位。该选项用于使该路由器首选从其他接口得到的 HELLO 路由（相比通过指定接口得到的 HELLO 路由）。

metricout *metric*
指定要添加到通过指定接口发送的路由的 HELLO 度量单位。缺省值为零。该选项用于使其他路由器首选优先该路由器的 HELLO 路由的其他来源。

trustedgateways *gateway_list*
定义 HELLO 将接受其中的更新的网关列表。 *gateway_list* 只是主机名或 IP 地址的列表。缺省情况下，共享网络上的所有路由器都受到信任，可提供路由信息。但是如果指定了 **trustedgateways** 子句，则仅接受来自列表中网关的更新。

sourcegateways *gateway_list*
定义 HELLO 向其直接（而不是通过多播或广播）发送数据包的路由器列表。它可用于向特定网关发送不同的路由信息。对该列表中网关的更新不受接口上 **noripout** 的影响。

traceoptions *trace_options*
指定 HELLO 的跟踪选项。（请参阅下面的跟踪语句和 HELLO 特定跟踪选项）。

缺省优先级是 90。缺省度量单位是 30000。

跟踪选项

每当声明新路由，所声明的度量单位更改或者路由进入或离开锁禁状态时，**policy** 选项将记录相应信息。

数据包跟踪选项（可以使用 **detail**、**send** 和（或）**recv** 修改）：

packets

所有 HELLO 数据包

OSPF 协议

开放式最短路径路由 (OSPF) 是最短路径优先 (SPF) 或链接状态协议。OSPF 是一种内部网关协议，它在单个自治系统中的路由器之间分发路由信息。OSPF 选择成本最低的路径作为最佳路径。OSPF 适用于包含大量路由器的复杂网络，它提供了等同成本多路径路由，在这种路由中，单个目标的数据包可以通过多个接口同时发送。在链接状态协议中，每个路由器均维护一个数据库，它描述通过所收集的所有路由器的链接状态公告构建的整个 AS 拓扑结构。每个参与的路由器均通过 AS 以泛流的方式分发其本地状态（路由器的可用接口和可连接的邻居）。每个包含至少两个路由器的多路访问网络都具有一个指定路由器和一个备用指定路由器。指定路由器将泛流多路访问网络的链接状态公告，并承担其他特殊责任。指定路由器的概念减少了多路访问网络上所需的相邻路由器的数量。

OSPF 允许将网络分成多个区域。将归纳在区域之间传递的路由信息，从而可以大大减少路由信息量。OSPF 使用四种不同的路由，下面按优先级顺序列出：区域内、区域间、类型 1 外部和类型 2 外部。区域内路径具有相同区域内的目标，区域间路径具有其他 OSPF 区域中的目标，自治系统外部 (ASE) 路由是 AS 外部目标的路由。作为类型 1 路由导入 OSPF 的路由应该来自 IGP，其外部度量单位可以与 OSPF 度量单位直接比较。在做出路由决定时， OSPF 会将 AS 边界路由器的内部成本添加到外部度量单位。类型 2 ASE 用于EGP，其度量单位与 OSPF 度量单位无法比较。这种情况下，路由决定中仅使用 AS 边界路由器的内部 OSPF 成本。

在拓扑数据库中，每个路由器均构成以其自身作为根的最短路径树。该最短路径树向
AS 中每个目标提供路由。外部派生的路由信息在该树显示为叶片。链接状态公告格式区分从外部来源获取的信息和从内部来源获取的信息，因此对于路由的来源和可靠性不存在任何疑问。外部派生的路由信息（如从 EGP 或 BGP 获取的路由）以透明方式通过自治系统进行传递，并且与 OSPF 内部派生的数据分开保存。每个外部路由也可以由公告路由器来添加标记，从而可以在自治系统边界的路由器之间传递其他信息。

OSPF 选择性地包括服务类型 (TOS) 路由，并允许管理员为指定目标的各种类型服务（低延迟或高吞吐量）安装多个路由。运行 OSPF 的路由器使用目标地址和服务类型来选择目标的最佳路由。

OSPF 区域内和区域间路由始终以优先级 10 导入 GateD 路由数据库。如果 OSPF 路由器不完全参与该区域的 OSPF，则将违反协议，因此不能将其覆盖。虽然可以显式地为其他路由提供更低的优先级值，但不建议这样做。

此外，还将尽可能地使用硬件多播功能，以提供链接状态消息。 OSPF 区域通过主干区域（即标识符为 0.0.0.0 的区域）进行连接。所有区域必须在逻辑上连续，主干也不例外。为了提供最大的灵活性，OSPF 允许配置虚拟链接，使得主干区域看起来是连续的（虽然实际上并非如此）。

一个区域中的所有路由器必须与该区域的参数一致。对于每个区域将单独运行一份链接状态算法。因此，大多数配置参数将逐个区域地进行定义。属于一个区域的所有路由器必须对该区域的配置一致。错误配置可能导致邻居之间无法形成邻接，以及路由信息可能不流动，甚至产生循环。

身份验证

对所有 OSPF 协议交换均将进行身份验证。身份验证确保仅从受信任的路由器导入路由信息，以保护 Internet 及其用户。可以使用的身份验证方案多种多样，但是必须为每个区域配置一个方案。这样，某些区域可以使用比其他区域严格得多的身份验证。 OSPF 协议交换可以进行身份验证。身份验证确保仅从受信任的路由器导入路由信

息，以保护 Internet 及其用户。可以使用两种身份验证方案。第一种使用包含最多 8 个字符的简单身份验证密钥，该方案已经过标准化。第二种仍处于试验阶段，它使用 MD5 算法和包含最多 16 个字符的身份验证密钥。

由于在许多情况下均可以轻易地从网络捕获数据包并获取身份验证密钥，因此简单口令提供了非常少的保护。试验阶段的 MD5 算法不会将身份验证密钥包含在数据包中，因此它能提供更多的保护。

OSPF 规范当前规定身份验证类型必须逐个区域地进行配置，并且支持给每个接口配置不同的口令。这已经扩展到允许给每个接口配置不同的身份验证类型和密钥。此外，可以在每个接口上同时指定主验证类型和密钥和二级身份验证类型和密钥。外发数据包使用主身份验证类型，但传入数据包可以匹配主验证类型和密钥或二级身份验证类型和密钥。

OSPF 语句

ospf yes | **no** | **on** | **off** [**{**
  defaults {
    preference *preference* **;**
    cost *cost* **;**
    tag [**as**] *tag* **;**
    type *1* | *2* **;**
  } ;
  exportlimit *routes* **;**
  exportinterval *time* **;**
  traceoptions *trace_options* **;**
  monitorauthkey *authkey* **;**
  monitorauth none | ([**simple** | **md5**] *authkey*) **;**
  backbone | (**area** *area*) **{**
    authtype 0 | **1** | **none** | **simple ;**
    stub [**cost** *cost*] **;**
    networks {
      network [**restrict**] **;**
      network **mask** *mask* [**restrict**] **;**
      network **masklen** *number* [**restrict**] **;**
      host *host* [**restrict**] **;**
    } ;
    stubhosts {
      host **cost** *cost* **;**
    } ;
    interface *interface_list*; [**cost** *cost*] **{**
      interface_parameters
    } ;
    interface *interface_list* **nonbroadcast** [**cost** *cost*] **{**

pollinterval *time* **;**
routers {
gateway [**eligible**] **;**
} ;
interface_parameters
} ;
仅限主干:
virtuallink neighborid *router_id* **transitarea** *area* **{**
interface_parameters
} ;
} ;
}] ;

下面是上面提及的 *interface_parameters*。它们可以在任何接口类上指定，并在 **interface** 子句下进行说明。

enable | disable ;
retransmitinterval *time* **;**
transitdelay *time* **;**
priority *priority* **;**
hellointerval *time* **;**
routerdeadinterval *time* **;**
authkey *auth_key* **;**

defaults

这些参数指定在将 OSPF ASE 路由导入 GateD 路由表和将路由从 GateD 路由表导出至 OSPF ASE 时使用的缺省值。

preference *preference*

优先级用于确定 OSPF 路由如何与来自 GateD 路由表中其他协议的路由进行竞争。缺省值为 150。

cost *cost*

在将非 OSPF 路由从 GateD 路由表作为 ASE 导出至 OSPF 时，将使用成本。缺省值是 1。可以在导出策略中将其显式覆盖。

tag [**as**] *tag*

OSPF ASE 路由包含一个 32 位标志字段，它不由 OSPF 协议使用，但可以由导出策略用来过滤路由。OSPF 与 EGP 进行交互时，可以使用该标记字段来传播 AS 路径信息，这种情况下将指定 **as** 关键字，并且标记限制为 12 位信息。如果未指定，则该标记设置为零。

type *1* | *2*

从 GateD 路由表导出至 OSPF 的路由缺省为类型 1 ASE。可以在此处显式更改该缺省值，并且可以在导出策略中覆盖。

ASE export rate

由于 OSPF 的性质，必须限制 ASE 的泛流率。这两个参数可用于调整这些速率限制。

exportinterval *time*

它指定生成一批 ASE 链接状态公告并将其泛流到 OSPF 的频率。缺省值是每秒一次。

exportlimit *routes*

该参数指定在每一批中将生成和泛流的 ASE 的数量。缺省值是 100。

traceoptions *trace_options*

指定 OSPF 的跟踪选项。（请参阅下面的跟踪语句和 OSPF 特定跟踪选项）。

monitorauthkey *authkey*

可以使用 **ospf_monitor** （它应该是超链接）实用程序查询 OSPF 状态。该实用程序发送非标准 OSPF 数据包，后者生成 OSPF 的文本响应。缺省情况下，这些请求未进行身份验证，但如果配置了身份验证密钥，则传入请求必须匹配指定的身份验证密钥。这些数据包不能更改任何 OSPF 状态，但查询 OSPF 的操作可以利用系统资源。

backbone

area *area*

每个 OSPF 路由器必须配置到至少一个 OSPF 区域中。如果配置了多个区域，则至少一个必须是 **backbone**。主干只能使用 **backbone** 关键字来配置，它不能指定为 **area 0**。主干接口可以是 **virtuallink**。

authtype 0 | 1 | none | simple

OSPF 给每个区域指定一个身份验证方案。即使区域中的每个接口可以使用不同的 **authenticationkey**，仍必须使用这一相同的身份验证方案。当前的有效值为 **none (0)**（无身份验证）或 **simple (1)** （简单口令身份验证）。

stub [**cost** *cost*]

stub 区域是不存在任何 ASE 路由的区域。如果指定了 **cost**，则它用于以指定成本将缺省路由插入该区域。

networks

networks 列表描述了区域的范围。处于指定范围的区域内 LSA 不会公告到其他区域作为区域间路由。指定的范围将公告为摘要网络 LSA。如果指定了 **restrict**，则不公告摘要网络 LSA。不处于任何范围的区域内 LSA 也将公告为摘要网络 LSA。该选项在设计良好的网络上非常有用，它可减少在区域间传播的路由信息量。该列表中的条目可以是网络或子网/掩码对。有关指定范围的详细信息，请参阅“路由过滤”一节。

stubhosts

该列表指定应公告为通过该路由器可连接的直接连接主机，以及公告时应该使用的成本。在此处应指定在其上不应运行 OSPF 的点对点接口。

下面的方法也很有用：给环回接口（不在 127 网络上的接口）分配一个附加地址，并将其公告为 stub 主机。如果该地址是用作路由器 ID 的相同地址，则允许通过路由器 ID （而不是通过接口地

址）路由到 OSPF 路由器。这比路由到可能始终无法连接的其中一个路由器接口地址更可靠。

interface *interface_list* [**cost** *cost*]

该格式的接口子句用于配置 **broadcast**（需要 IP 多播支持）或 **point-to-point** 接口。有关 *interface_list* 的说明，请参阅有关接口列表规范的部分。

每个接口都具有一个 **cost**。将到达某个目标之前必须经过的所有接口的成本累加起来，以得到该目标的成本。缺省成本是一，但也可以指定其他非零值。

所有接口类型均通用的接口参数包括：

retransmitinterval *time*

为属于该接口的邻接重新传输连接状态公告的间隔秒数。

transitdelay *time*

通过该接口传输链接状态更新所需的估计秒数。 transitdelay 将计入传输和传播延迟，并且必须大于 0。

priority *priority*

介于 0 和 255 之间的数字，指定成为该接口上指定路由器的优先级。当网络上连接的两个路由器同时尝试成为指定路由器时，将首选具有最高优先级的路由器。其路由器优先级设置为 0 的路由器不具备成为指定路由器的资格。

hellointerval *time*

路由器在接口上发送 Hello 数据包之间间隔的时间长度（秒）。

routerdeadinterval *time*

在路由器的邻居声明路由器关闭之前未收到该路由器的 Hello 数据包的秒钟数。

authkey *auth_key*

由 OSPF 身份验证用来生成并验证 OSPF 标头中的身份验证字段。可以逐个接口地配置身份验证密钥。它由一个到八个由句点分隔的十进制位、前加 **0x** 的一到八字节的十六进制字符串或者双引号内的一至八个字符的字符串指定。

点对点接口还支持该附加参数：

nomulticast

缺省情况下，点对点接口上邻居的 OSPF 数据包通过 IP 多播机制发送。但是，Unix 的某些 IP 多播实现具有缺陷，阻止在这些接口上使用 IP 多播。 GateD 将检测该情况，并恢复到采用将单播 OSPF 数据包发送到该点对点邻居的方法。

如果由于远程邻居不支持而不应使用 IP 多播，则可以指定 **nomulticast** 参数来强制使用单播 OSPF 数据包。该选项也可用于去除在 Gated 检测到上述缺陷时发出的警告。

interface *interface_list* **nonbroadcast** [**cost** *cost*]

这种格式的 interface 子句用于指定 **nonbroadcast** 多路访问 (NBMA) 介质上的 **nonbroadcast** 接口。由于 OSPF **broadcast** 介质必须支持 IP 多播，因此必须将不支持 IP 多播的广播介质（如以太网）配置为非广播接口。

非广播介质支持上面列出的任何标准 **interface** 子句以及下面两个针对非广播接口的子句：

pollinterval *time*

在与邻居建立邻接关系之前，将以指定的 **pollinterval** 定期发送 OSPF 数据包。

routers 根据定义，不可能通过发送广播数据包来发现非广播接口上的 OSPF 邻居，因此必须配置所有邻居。该列表包括一个或多个邻居，并指明它们是否具备成为指定路由器的资格。

virtuallink neighborid *router_id* **transitarea** *area*

虚拟链接用于建立或增加主干区域的连接。**neighborid** 是虚拟链接另一端的 *router_id*。必须在该系统上配置传输区域 (**area**)。可以在虚拟链接上指定以上 **interface** 子句所定义的所有标准接口参数。

跟踪选项

除了以下 OSPF 特定跟踪标志之外，OSPF 还支持 **state**，它跟踪接口和邻居状态机转换。

lsabuild

链接状态公告创建

spf 最短路径优先 (SPF) 计算

数据包跟踪选项（可以用 **detail**、**send** 和 **recv** 修改）：

hello 可用于确定邻居连接性的 OSPF **HELLO** 数据包。

dd 用于同步 OSPF 数据库的 OSPF 数据库说明数据包。

request

用于同步 OSPF 数据库的 OSPF 链接状态请求数据包。

lsu 用于同步 OSPF 数据库的 OSPF 链接状态更新数据包。

ack 用于同步 OSPF 数据库的 OSPF 链接状态确认数据包。

外部网关协议 (EGP)

外部网关协议 (EGP) 是用于与其他自治系统中的网关交换路由信息的外部路由协议。与内部协议不同，EGP 仅传播连接性指示，而不是实际的度量单位。EGP 更新包含度量单位，称作距离，其范围是 0 到 255。GateD 将仅比较从相同 AS 得到的 EGP 距离。

EGP 将路由信息发送到远程路由器之前，它必须建立与该路由器的邻接关系。这通过与该路由器交换 *Hello*（不要与 HELLO 协议或 OSPF HELLO 消息混淆）和 *I Heard You* (I-H-U) 消息来实现。通过 EGP 通信的计算机称作 EGP 邻居，HELLO 和 I-H-U 消息交换称作获取邻居。获取邻居后，系统将轮询邻居以获取路由信息。邻居通过

发送包含路由信息的更新来进行响应。如果系统从其邻居收到轮询，它将以其自己的更新数据包进行响应。当系统收到更新时，它会将更新中的路由包括到其路由数据库中。如果邻居未能响应三次连续轮询， GateD 将假定邻居已关闭，并从其数据库中删除该邻居的路由。

EGP 语句

egp yes | no | on | off
[{
  preference *preference* **;**
  defaultmetric *metric* **;**
  packetsize *number* **;**
  traceoptions *trace_options* **;**
  group
    [**peeras** *autonomous_system*]
    [**localas** *autonomous_system*]
    [**maxup** *number*]
  {
    neighbor *host*
      [**metricout** *metric*]
      [**preference** *preference*]
      [**preference2** *preference*]
      [**ttl** *ttl*]
      [**nogendefault**]
      [**importdefault**]
      [**exportdefault**]
      [**gateway** *gateway*]
      [**lcladdr** *local_address*]
      [**sourcenet** *network*]
      [**minhello | p1** *time*]
      [**minpoll | p2** *time*]
      [**traceoptions** *trace_options*]
        ;
  } ;
}] ;

preference *preference*

设置从 RIP 得到的路由的优先级。缺省优先级是 200。该优先级可以用 **group** 或 **neighbor** 语句中指定的优先级或导入策略来覆盖。

defaultmetric *metric* **;**

定义在通过 EGP 公告路由时使用的度量单位。如果未指定，则缺省值为 255，某些系统可能会认为它无法连接。这种值选择要求您在将路由导出至 EGP 邻居时显式指定度量单位。该度量单位可以用 neighbor 或 group 语句或导出策略中指定的度量单位来覆盖。

packetsize *maxpacketsize*

它定义 EGP 希望从该邻居接收的数据包的最大预期大小。如果收到大于该值的数据包，它将是不完整的，必须忽略。将记下该数据包的长度，然后将增加预期大小，以能够接收该大小的数据包。在此处指定参数将防止丢弃第一个数据包。如果未指定，则缺省大小为 8192 个字节。所有数据包大小都将四舍五入为系统页大小的倍数。

traceoptions *trace_options*

指定 EGP 的跟踪选项。缺省情况下，从全局跟踪选项继承它们。这些值可以按照组或邻居进行覆盖（请参阅下面的跟踪语句和 EGP 特定跟踪选项）。

group EGP 邻居必须指定为 **group** 的成员。组通常用于将所有邻居归入一个自治系统中。除非在 neighbor 子句上显式覆盖，否则在 group 子句上指定的参数将适用于所有辅助邻居。任意数量的 **group** 子句可以指定任意数量的 **neighbor** 子句。

neighbor 次子句中的任何参数可以在 **group** 子句上指定，以提供整个组的缺省值（可以为单个邻居覆盖）。此外，**group** 子句是可设置以下属性的唯一子句：

peeras 标识从组中对等端预期的自治系统数。如果未指定，则将动态获取。

localas 标识 GateD 要提供给组的自治系统。缺省值是在 **autonomoussystem** 语句中全局设置的值。通常只有在充当其他自治系统时才使用该选项，其他情况下不建议使用。

maxup 指定 GateD 应该从该组中获取的邻居数。缺省为获取组中的所有邻居。GateD 将尝试按列出的顺序获取前 **maxup** 个邻居。如果这些邻居中的某一个不可用，则将获取列表中的下一个邻居。如果 GateD 在启动后确实设法获取最适合的邻居，则将丢失较不适合的邻居。

neighbor *neighbor_address*

每个 neighbor 子句在组中定义一个 EGP 邻居。次子句的唯一必需部分是 **neighbor_address** 参数，它是邻居的符号主机名或 IP 地址。其他所有参数均是可选的。

preference *preference*

指定用于从这些邻居获取的路由的优先级。它可能不同于在 **egp** 语句中设置的缺省 EGP 优先级，因此 GateD 可以首选优先于邻居或邻居组中其他路由的路由。该优先级可以由导入策略显式覆盖。

preference2 *preference*

对于 **preference** 等同的情况，可能使用第二个优先级 **preference2** 来进行仲裁。缺省值为 0。

metricout *metric*

它定义要用于发送到该邻居的所有路由的度量单位。该值覆盖 **egp** 语句中设置的缺省度量单位和由导出策略指定的任何度量单位（但仅适用于该特定邻居或邻居组）。

nogendefault

防止 GateD 在 EGP 从其邻居收到有效更新时生成缺省路由。缺省路由仅在启用 gen-default 选项时生成。

importdefault

可使 GateD 接受缺省路由 (0.0.0.0) （如果它包含在收到的 EGP 更新中）。如果未指定，则忽略 EGP 更新中包含的缺省路由。为了提高效率，某些网络让外部路由器声明缺省路由，以避免发送较大的 EGP 更新数据包。

exportdefault

可使 GateD 将缺省路由 (0.0.0.0) 包含在发送给该 EGP 邻居的 EGP 更新中。这样，系统就可以通过 EGP 公告缺省路由。通常，缺省路由不包含在 EGP 更新中。

gateway *gateway*

如果网络没有与邻居共享，则 **gateway** 会指定将连接网络上的路由器用作从该邻居接收的路由的下一跃点路由器。该选项仅在极少的情况下使用。

lcladdr *local_address*

指定将在邻居连接的本地端使用的地址。本地地址必须位于与邻居或邻居的网关（如果使用 **gateway** 参数）共享的接口上。只有当具有适当本地地址（通过它可直接连接邻居或网关地址）的接口正在运行时，才会打开会话。

sourcenet *network*

指定在 EGP 轮询数据包中查询的网络。缺省情况下，它是与指定地址的邻居共享的网络。如果没有网络与该邻居共享，则应该指定该邻居连接到的网络之一。该参数也可用于指定与用来发送 EGP 数据包的邻居之外的邻居进行共享的网络。通常不需要该参数。

p1 *time*

minhello *time*

设置 EGP *HELLO* 数据包传输之间可接受的最小时间间隔。缺省的 hello 时间间隔是 30 秒。如果邻居未能响应三个 hello 数据包， GateD 将停止尝试获取邻居。通过设置较大的时间间隔，可以让邻居有更大的机会来进行响应。Minhello 是 EGP 规范中定义的 **P1** 值的别名。

p2 *time*

minpoll *time*

设置邻居轮询之间的时间间隔。缺省值为 120 秒。如果发送三个轮询而未收到响应，则将声明该邻居已“关闭”，并从路由数据库中删除从该邻居获取的所有路由。较长的轮询间隔支持更为稳定的路由数据库，但是它对路由更改的响应会减缓。 Minpoll

是 EGP 规范中定义的 **P2** 值的别名。

ttl *ttl* 缺省情况下，GateD 将本地邻居的 IP TTL 设置为 *1*，将非本地邻居的 TTL 设置为 255。当尝试与忽略以值为 1 的 TTL 发送的数据包的错误运行路由器进行通信时，将提供该选项。

traceoptions *trace_options*

指定该 EGP 邻居的跟踪选项。缺省情况下，从组或 EGP 全局跟踪选项继承它们。（请参阅下面的跟踪语句和 EGP 特定跟踪选项）。

跟踪选项

state 和 **policy** 选项与 EGP 一起使用。

数据包跟踪选项（可以用 **detail**、 **send** 和 **recv** 修改）：

packets

所有 EGP 数据包

hello 可用于确定邻居连接性的 EGP **HELLO/I-HEARD-U** 数据包。

acquire

用于启动和终止 EGP 会话的 EGP **ACQUIRE/CEASE** 数据包。

update 用于请求和接收连接性更新的 EGP **POLL/UPDATE** 数据包。

BGP 协议

BGP（Border Gateway Protocol，边界网关协议）是用于在自治系统之间交换路由信息的外部路由协议。 BGP 用于在多个传输自治系统之间以及在传输和存根自治系统之间交换路由信息。BGP 与 EGP 相关，但是运行时具有更多的功能、更大的灵活性，且需要的带宽更少。BGP 使用路径属性来提供每个路由的详细信息，并特别维护一个 *AS* 路径，它包括路由已传输的每个自治系统的 AS 编号，从而提供足够的信息来防止任意拓扑结构中出现路由循环。路径属性也可用于区分不同的路由组，以确定管理优先级，这样可以更加灵活地确定路由优先级，从而实现多种管理目的。

BGP 支持邻居之间的两种基本会话：内部（有时称作 IBGP）和外部。内部会话在同一个自治系统中的路由器之间运行，而外部会话则在不同自治系统中的路由器之间运行。向外部对等端发送路由时，本地 AS 编号将添加到 AS 路径之前，因此可确保从外部收到的路由具有该对等端在路径开头的 AS 编号。从内部邻居收到的路由通常不会将本地 AS 编号添加到 AS 路径之前，因此通常具有路由在源内部邻居从外部对等端收到路由时所具有的相同 AS 路径。可以从内部邻居合法接收路径中无 AS 编号的路由；这些路由指示应该将收到的路由视为您自己的 AS 的内部路由。

BGP 实现支持三个版本的 BGP 协议：版本 2、3 和 4。BGP 版本 2 和 3 在功能方面非常相似。它们仅传播分类的网络路由，而 AS 路径是 AS 编号的简单数组。BGP 4 将传播具有完全通用的地址-掩码路由，AS 路径具有某种结构来表示汇聚非类似路由的结果。

外部 BGP 会话可能会也可能不会包括单个度量单位， BGP 在路径属性中将其称作多出口标识。对于 BGP 版本 2 和 3，该度量单位是 16 位无符号整数，对于 BGP 版本 4，它是 32 位无符号整数。在任一种情况下，均将首选度

量单位的较小值。目前，该度量单位仅用于在来自相同邻居 AS 且具有相同优先级的路由之间仲裁。内部 BGP 会话在路径属性中带有至少一个度量 ノ唬?GP 将其称作 *LocalPref*。该度量单位的大小等于 MED。对于 BGP 版本 2 和 3，认为该度量单位越小就越好，对于版本 4，越大就越好。BGP 版本 4 会话可以选择在内部会话上带有另一个度量单位，这是多出口标识的内部版本。这些度量单位的使用取决于指定的内部协议处理类型。

BGP 将带有类似路径属性的路由折叠到单个更新中进行公告。在单个更新中收到的路由将在单个更新中重新声明。这将尽量减小因丢失邻居而造成的影响，并且将在最大程度上压缩在建立对等端的过程中发送的初始公告。BGP 不从内核中逐条消息地读取信息，但会填充输入缓冲区。它首先处理缓冲区中的所有完整消息，然后再次读取。BGP 还将执行多次读取，以清除套接字上列队的所有传入数据。该功能可能导致繁忙的对等端连接在较长的时间间隔内阻塞其他协议。

所有不可达的消息均收集到单个消息中，在瞬时更新中的可达路由之前发送。对于这些无法连接的声明，下一跃点将设置为连接上的本地地址，而不发送任何度量单位，并且路径来源将设置为不完整。在外部连接上，无法连接声明中的 AS 路径设置为本地 AS；在内部连接上， AS 路径设置为零长度。

BGP 实现期望外部对等端直接连接到共享子网上，并期望这些对等端声明该子网上主机地址的下一跃点（虽然可以通过用于测试的配置来放宽该限制）。但是对于内部对等端组，存在多个选项，通过指定组类型即可从中进行选择。类型为 **internal** 的组期望所有对等端均直接连接到共享子网，以便像外部对等端一样，在 BGP 公告中收到的下一跃点可直接用于转发。类型为 **routing** 的组将使用以对等端中的路由作为转发地址而接收的下一跃点来确定路由的直接下一跃点。该转发地址用于在 IGP 的路由中查找直接下一跃点。这些组支持远程对等端，但需要知道它们使用其路由来确定下一跃点的 IGP。最后，类型为 **igp** 的组期望组对等端的路由根本不用于转发。它们期望收到的 BGP 路由副本还将通过 IGP 接收，并且 BGP 路由将仅用于确定与 IGP 路由关联的路径属性。这些组也支持远程对等端，并且也需要知道它们运行时所使用的 IGP。

对于内部 BGP 组类型（并且对于测试组），将尽可能基于公共策略为所有组对等端构建单个外发消息。该消息的一个副本将发送给组中的每个对等端，并且可能根据每个对等端的具体情况对下一跃点字段进行调整。这将尽量减少在这些类型的组中运行大量对等端时的计算量。如果已经用 **allow** 子句对相应的组进行了配置，则 BGP 将允许未配置的对等端建立连接。

BGP 语句

bgp yes | no | on | off
[{
  preference *preference* **;**
  defaultmetric *metric* **;**
  traceoptions *trace_options* **;**
  group type (**external peeras** *autonomous_system*)
    | (**internal peeras** *autonomous_system*)
    | (**igp peeras** *autonomous_system* **proto** *proto*)
    | (**routing peeras** *autonomous_system* **proto** *proto*
        interface *interface_list*)
    | (**test peeras** *autonomous_system*)
  {

allow {
network
network **mask** *mask*
network **masklen** *number*
all
host *host*
} ;
peer *host*
[**metricout** *metric*]
[**localas** *autonomous_system*]
[**nogendefault**]
[**gateway** *gateway*]
[**preference** *preference*]
[**preference2** *preference*]
[**lcladdr** *local_address*]
[**holdtime** *time*]
[**version** *number*]
[**passive**]
[**sendbuffer** *number*]
[**recvbuffer** *number*]
[**indelay** *time*]
[**outdelay** *time*]
[**keep** [**all** | **none**]]
[**showwarnings**]
[**noauthcheck**]
[**noaggregatorid**]
[**keepalivesalways**]
[**v3asloopokay**]
[**nov4asloop**]
[**logupdown**]
[**ttl** *ttl*]
[**traceoptions** *trace_options*]
;
} ;
}] ;

external | internal | igp | test

bgp 语句可启用或禁用 BGP。在缺省情况下将禁用 BGP。通过 BGP 声明路由的缺省度量单位将不发送度量单位。

preference *preference*

设置从 RIP 得到的路由的优先级。缺省优先级是 170。该优先级可以用 **group** 或 **peer** 语句中指定的优先级或导入策略来覆盖。

defaultmetric *metric*

定义在通过 BGP 公告路由时使用的度量单位。如果未指定，则不传播度量单位。该度量单位可以用 neighbor 或 group 语句或导出策略中指定的度量单位来覆盖。

traceoptions *trace_options*

指定 BGP 的跟踪选项。缺省情况下，从全局跟踪选项继承它们。这些值可以按照组或邻居进行覆盖（请参阅下面的跟踪语句和 BGP 特定跟踪选项）。

组

BGP 对等端按照类型和对等端的自治系统进行分组。可以指定任意数量的组，但是每个组必须具有类型和对等端自治系统的唯一组合。下面是四种可能的组类型：

group type external peeras *autonomous_system*

在传统的外部 BGP 组中，完全策略检查适用于所有传入和外发的公告。外部邻居必须可以通过计算机的本地接口之一直接连接。缺省情况下，外部声明中不包含度量单位，并相对于共享接口来计算下一跃点。

group type internal peeras *autonomous_system*

在其中没有 IP 级 IGP 的网络（如 SMDS 网络或 MILNET）中运行的内部组。该组中的所有邻居必须可以通过单个接口直接连接。所有下一跃点信息将相对于该接口计算。导入和导出策略可应用于组公告。缺省情况下，从外部 BGP 或 EGP 邻居收到的路由将以收到的度量单位重新公告。

group type igp peeras *autonomous_system* **proto** *proto*

与内部协议关联运行的内部组。只有当无法用 IGP 标记机制完全表示路径属性时， IGP 组才会检查 IGP 所导出的路由并发送公告。只有 AS 路径、路径来源和过渡可选属性才随路径一起发送。将不发送度量单位，而下一跃点将设置为连接使用的本地地址。收到的内部 BGP 路由将不用于重新声明。不过，AS 路径信息将附加到相应的 IGP 路由，后者将用于重新公告。由于仅向内部 IGP 对等端发送 IGP 所导出的路由的子集，因此将使用 IGP 的导出策略。由于声明的路由是 IGP 已导出的路由，因此无须实现“don’t routes from peers in the same group”限制。

group type routing peeras *autonomous_system* **proto** *proto* **interface** *interface_list*

使用内部协议的路由解析转发地址的内部组。类型为 routing 的组在非直接连接的路由器之间传播外部路由。类型为 routing 的组使用以路由作为转发地址到达的 BGP 下一跃点计算这些路由的直接下一跃点。转发地址通过内部协议的路由信息来进行解析。实际上，内部 BGP 用于传送 AS 外部路由，而 IGP 将仅传送 AS 内部路由，后者用于查找前者的直接下一跃点。

proto 指定要用于解析 BGP 路由下一跃点的内部协议，并且可以是配置中任意 IGP 的名称。缺省情况下，因为假定该地址的路由将通过 IGP 传播，因此向类型为 routing 的对等端声明的 BGP 路由中的下一跃点将设置为这些对等端的 BGP 连接上的本地地址。 *interface_list* 可以选择提供接口列表，这些接口的路由通过改用第三方下一跃点的 IGP 来传送。

group type test peeras *autonomous_system*

使用测试对等端实现固定策略的外部 BGP 扩展。固定策略和特殊情况代码将相对降低测试对等端的维护费用。测试对等端不需要位于直接连接的网络上。如果 GateD 和对等端位于相同的（直接连接）子网上，则将相对于该网络计算公告的下一跃点。否则，下一跃点将是本地计算机的当前下一跃点。将忽略由测试对等端公告以及从其中接收的所有路由信息，可公告的所有 BGP 路由均将发送回测试对等端。 EGP 派生和 BGP 派生路由的度量单位在声明中转发。否则将不包含任何度量单位。

组参数

BGP 语句包括 **group** 子句和 **peer** 次子句。在一个组中可以指定任意数量的对等端次子句。group 子句通常定义对等端组的缺省参数，这些参数适用于所有附属对等端次子句。 peer 次子句中的任何参数均可以在 group 子句上指定，以提供整个组的缺省值（可以为单个对等端覆盖）。

指定对等端

在一个组中，可以通过两种方式之一配置 BGP 对等端。它们可以使用 **peer** 语句显式配置，也可以使用 **allow** 语句隐式配置。下面将介绍这两种方式：

allow allow 子句允许从指定范围的网络和掩码对中的任何地址建立 **peer** 连接。这些对等端的所有参数必须在 group 子句上配置。内部对等端结构在收到传入打开请求时创建，在断开连接时删除。有关指定网络和掩码对的详细信息，请参阅“路由过滤”一节。

peer *host*

peer 子句配置单个对等端。每个对等端继承在 group 子句上指定为缺省值的所有参数。这些缺省值可以由 peer 次子句上指定的参数显式覆盖。

在每个 **group** 子句中，可以指定单个对等端，也可以使用 **allow** 指定一组可能的对等端。 **Allow** 用于指定一组地址掩码。如果 GateD 从指定集合中的任意地址收到 BGP 连接请求，则将接受该请求并建立对等端关系。

对等端参数

BGP **peer** 次子句支持以下参数，它们也可在 **group** 子句上指定。所有参数均是可选的。

metricout *metric*

如果指定，则该度量单位将用作发送到指定对等端上的所有路由的主度量单位。该度量单位覆盖缺省度量单位，即在 group 子句上指定的度量单位以及导出策略指定的任意度量单位。

localas *autonomous_system*

确定 GateD 要提供给该对等端组的自治系统。缺省值是在 **autonomoussystem** 语句中全局设置的值。

nogendefault

防止 GateD 在 EGP 从其邻居收到有效更新时生成缺省路由。缺省路由仅在启用 gendefault 选项时生成。

gateway *gateway*

如果网络没有与对等端共享，则 **gateway** 会指定将连接网络上的路由器用作从该邻居接收的路由的下一跃点路由器。大多数情况下不需要该参数。

preference *preference*

指定用于从这些对等端获取的路由的优先级。它可能不同于在 **bgp** 语句中设置的缺省 BGP 优先级，因此 GateD 可以首选优先于对等端或对等端组中其他路由的路由。该优先级可以由导入策略显式覆盖。

preference2 *preference*

对于 **preference** 等同的情况，可能使用第二个优先级 **preference2** 来进行仲裁。缺省值为 0。

lcladdr *local_address*

指定将在对等端 TCP 连接的本地端使用的地址。对于外部对等端，本地地址必须位于与邻居或邻居的网关（如果使用 **gateway** 参数）共享的接口上。只有当具有适当本地地址（通过它可直接连接对等端或网关地址）的接口正在运行时，才会打开包含外部对等端的会话。对于其他类型的对等端，当包含指定本地地址的任何接口正在运行时，将维护对等端会话。在任一情况下，只有在传入的连接寻址到配置的本地地址时，才将其识别为匹配已配置的对等端。

holdtime *time*

指定与该对等端协商连接时使用的 BGP 保留时间值（秒）。根据 BGP，如果 GateD 在 BGP 打开消息的“保留时间”字段中指定的时间内没有收到保持活动、更新或通知消息，则将关闭该 BGP 连接。该值必须是零（不发送任何保持活动消息）或者至少是 3。

version *version*

指定要用于该对等端的 BGP 协议的版本。如果未指定，则将首先使用最高的支持版本，并尝试版本协商。如果已指定，则协商过程中仅提供指定的版本。目前支持的版本是 2、3 和 4。

passive 指定不应该尝试该对等端的活动 OPEN。 GateD 应该等待对等端发出 OPEN 消息。缺省情况下，所有显式配置的对等端将处于活动状态，在对等端响应之前，它们将定期发送 OPEN 消息。

sendbuffer *buffer_size*

recvbuffer *buffer_size*

控制向内核请求的发送和接收缓冲量。最多支持 65535 个字节（即使许多内核具有更低的限制）。缺省情况下，GateD 将配置支持的最大值。正常运行的系统上不需要这些参数。

indelay *time*

outdelay *time*

用于降低路由的不稳定性。**Indelay** 是将从 BGP 对等端获取的路由纳入 GateD 路由数据库之前必须保持稳定的时间。 **Outdelay** 是路由在导出至 BGP 之前必须在 GateD 路由数据库中存在的时间。其缺省值均为 0，表示禁用这些功能。

keep all

用于获取从对等端获取的路由（即使路由的 AS 路径包含导出的 AS 编号之一）。

showwarnings

可使 GateD 在收到可疑的 BGP 更新（如重复的路由和（或）删除不存在的路由）时发出警告消息。通常将以无提示方式忽略这些事件。

noauthcheck

通常，GateD 将验证是否传入的数据包包含均为一的验证字段。该选项可用于允许与使用其他形式身份验证的实现进行通信。

noaggregatorid

可使 GateD 将汇聚程序属性中的路由器 ID 指定为零（而不是其路由器 ID），以防止 AS 中的不同路由器用不同的 AS 路径创建汇聚路由。

keepalivesalways

可使 GateD 始终发送保持活动消息（即使可以正确地使用更新替换一）。它允许与目前不完全遵守协议规范的路由器进行相互操作。

v3asloopokay

缺省情况下，GateD 将不向版本 3 外部对等端声明其 AS 路径循环（AS 多次出现在该路径中）的路由。设置该标志可删除这一限制。在内部组或对等端上设置时将被忽略。

nov4asloop

防止向版本 4 外部对等端声明包含循环 AS 路径的路由。它可用于避免向会将路由错误转发到版本 3 邻居的对等端公告这些路由。

logupdown

可使每次 BGP 进入或离开 **ESTABLISHED** 状态时通过 syslog 机制记录消息。

ttl *ttl* 缺省情况下，GateD 将本地对等端的 IP TTL 设置为*1*，将非本地对等端的 TTL 设置为 255。该选项主要在尝试与运行不正常的路由器进行通信时使用，这些路由器会忽略以 TTL 为 1 发送的数据包。并非所有内核都允许为 TCP 连接指定 TTL。

traceoptions *trace_options*

指定该 BGP 邻居的跟踪选项。缺省情况下，将从组或 BGP 全局跟踪选项继承它们。（请参阅下面的跟踪语句和 BGP 特定跟踪选项）。

跟踪选项

请注意，**state** 选项可用于 BGP，但不提供实际状态转换信息。

数据包跟踪选项（可以用 **detail**、**send** 和 **recv** 修改）：

packets
所有 BGP 数据包

open 可用于建立对等端关系的 BGP **OPEN** 数据包。

update 用于传递网络连接性信息的 BGP **UPDATE** 数据包。

keepalive
可用于验证对等端连接性的 BGP **KEEPALIVE** 数据包。

ICMP 语句

在不包含 BSD 路由套接字的系统上，GateD 将监听系统收到的 ICMP 消息。当前 GateD 仅处理 ICMP 重定向数据包，但将来可能添加更多功能，例如对路由器发现消息的支持。ICMP 重定向消息的处理由 **redirect** 语句来处理。

目前指定 **icmp** 语句的唯一原因是能够跟踪 GateD 接收的 ICMP 消息。

ICMP 语句

icmp {
traceoptions *trace_options* **;**
}

traceoptions *trace_options* **;**
指定 ICMP 的跟踪选项（请参阅下面的跟踪语句和 ICMP 特定跟踪选项）。

跟踪选项

数据包跟踪选项（可以用 **detail** 和 **recv** 修改）：

packets
收到的所有 ICMP 数据包。

redirect
仅收到 ICMP **REDIRECT** 数据包。

routerdiscovery
仅收到 ICMP **ROUTER DISCOVERY** 数据包。

info 仅适用于 ICMP 信息性数据包，它包括掩码请求/响应、信息请求/响应、回显请求 /响应以及时间戳/响应。

error 仅适用于 ICMP 错误数据包，它们包括超过时间、参数问题、无法连接和源损坏。

重定向处理

通过监视 ICMP 消息或者通过支持它的系统上的路由套接字，将向 **redirect** 代码传递 ICMP 或 ISO 重定向。它处理重定向请求，并确定是否接受重定向。如果接受重定向，则将使用协议 **redirect** 将路由安装在 GateD 路由表中。3 分钟后，重定向将从路由表中删除。

如果 GateD 确定重定向不可接受，它将尝试确定是否已修改内核转发表。在监视 ICMP 消息的系统上，这通过再

次猜测内核要用重定向执行的操作来完成。在包含路由套接字的系统上，内核将提供是否接受重定向的指示；GateD 将忽略未处理的重定向。

如果 GateD 确定内核转发表的状态已更改，则将对内核发出必要的请求，以恢复正确的状态。

请注意，在当前可用的系统上，即使系统正在作为路由器运行，仍不可能禁止处理 ICMP 重定向。要忽略重定向的影响，GateD 必须处理每一个重定向，并且主动将做出的任何更改恢复到内核的状态。由于所涉及的机制，重定向的影响在一段时间内将存在于内核中。

缺省情况下，GateD 将在主动参与内部网关协议（RIP、 HELLO、OSPF 或 IS-IS）时删除重定向。重定向自动禁用之后，无法将其启用。无论是以非广播模式监听 RIP 或 HELLO，还是使用 EGP 和 BGP，均不会导致忽略重定向。在这些情况下必须通过手动配置来忽略重定向。

请注意，按照最新的“IETF 路由器要求”文档，GateD 将确保所有 ICMP 网络重定向均作为主机重定向进行处理。当接受 ICMP 网络重定向时，GateD 将向内核发出请求，确保更新内核转发表，以反映主机重定向而不是网络重定向。

重定向语句不防止系统发送重定向，而只防止系统监听重定向。

重定向语句

redirect yes | no | on | off
[{
  preference *preference* **;**
  interface *interface_list*
    [**noredirects**] | [**redirects**] **;**
  trustedgateways *gateway_list* **;**
  traceoptions *trace_options* **;**
}] **;**

preference
设置从重定向得到的路由的优先级。缺省值是 30。

interface *interface_list*
interface 语句允许逐个接口地启用和禁用重定向。有关 *interface_list* 的说明，请参阅有关接口列表规范的部分。可能的参数包括：

noredirects
指定将忽略通过指定接口接收的重定向。缺省为接受所有接口上的重定向。

redirects
这是缺省值。在通配符接口描述符上使用 **noredirects** 时，该参数是必需的。

trustedgateways *gateway_list*
定义将接受其中的重定向的网关列表。 *gateway_list* 只是主机名称或地址的列表。缺省情况下，共享网络上的所有路由器都受到信任，可提供重定向。但是如果指定了 **trustedgateways** 子句，

则仅接受来自列表中网关的重定向。

traceoptions *trace_options*

不存在重定向特定的跟踪选项。所有非错误消息均在 **normal** 类下跟踪。

跟踪选项

不存在 **Redirect** 特定的跟踪选项。所有非错误消息均在 **normal** 类下跟踪。

路由器发现协议

路由器发现协议是用于通知主机是否存在路由器的 IETF 标准协议。使用它可以不必具有主机 *wiretap* 路由协议（如 RIP）。它替换或附加到主机中静态配置的缺省路由使用。

该协议分为两个部分：服务器部分（在路由器上运行），客户端部分（在主机上运行）。 GateD 将它们当作两个单独的协议来进行处理，一次只能启用其中一个协议。

路由器发现服务器

路由器发现服务器在路由器上运行，并且向主机声明其存在状态。为此，它定期向启用该服务器的每个接口多播或广播 **Router Advertisement**。这些路由器公告包含给定接口上所有路由器地址的列表以及它们用作缺省路由器的优先级。

最初，这些路由器公告每隔几秒钟发生一次，然后恢复为每隔几分钟发生一次。此外，主机可以发送一个 **Router** 请求，路由器将使用单播路由器公告对其做出响应（除非多播或广播公告很快要到期）。

每个路由器公告包含一个公告生命周期字段，表示公告地址的有效期。该生命周期经过配置后，使得在生命周期过期之前将发送另一个路由器公告。零生命周期用于表示一个或多个地址不再有效。

在支持 IP 多播的系统上，路由器公告在缺省情况下将象所有主机多播地址 **224.0.0.1** 发送。但是，可以指定使用广播。当路由器公告发送到所有主机多播地址时，或者为有限广播地址 **255.255.255.255** 配置接口时，路由器公告中将包括物理接口上配置的所有 IP 地址。当路由器公告发送到网络或子网广播时，仅包含与该网络或子网关联的地址。

路由器发现服务器语句

routerdiscovery server yes | no | on | off [{
  traceoptions *trace_options* **;**
  interface *interface_list*
    [**minadvinterval** *time*]
    [**maxadvinterval** *time*]
    [**lifetime** *time*]
    ;
  address *interface_list*
    [**advertise**] | [**ignore**]
    [**broadcast**] | [**multicast**]
    [**ineligible**] | [**preference** *preference*]
    ;

```
}];
```

traceoptions *trace_options*

指定路由器发现跟踪选项（请参阅下面的跟踪语句和路由器发现特定跟踪选项）。

interface *interface_list*

指定适用于物理接口的参数。请注意，在惯例上与 GateD 的其他部分略有不同， **interface** 仅指定物理接口（如 **le0**、**ef0** 和 **en1**），而 **address** 则指定协议（这种情况下是 IP）地址。

接口参数包括：

maxadvinterval *time*

从接口在发送广播或多播路由器公告之间允许的最长时间。不得小于 *4* 且不得大于 *30:00*（30 分钟或 1800 秒）。缺省值是 **10:00**（10 分钟或 600 秒）。

minadvinterval *time*

从接口发送未经请求的广播或多播路由器公告之间允许的最短时间。不得小于 *3* 秒钟，且不得大于 **maxadvinterval**。缺省值是 **0.75 * maxadvinterval**。

lifetime *time*

路由器声明中地址的周期。不得小于 **maxadvinterval** 且不得大于 *2:30:00*（两小时三十分钟或 9000 秒）。缺省值是 **3 * maxadvinterval**。

address *interface_list*

指定适用于该物理接口上指定地址集的参数。请注意，在惯例上与 GateD 的其他部分略有不同，**interface** 仅指定物理接口（如 **le0**、 **ef0** 和 **en1**），而 **address** 则指定协议（这种情况下是 IP）地址。

advertise

指定路由器公告中应包括指定的地址。这是缺省值。

ignore 指定路由器公告中不应包括指定的地址。

broadcast

指定由于该系统不支持 IP 多播，或者连接网络上的某些主机不支持 IP 多播，广播路由器公告中应该包括给定的地址。可以在物理接口上混用地址，使某些地址包括在广播路由器公告中，某些地址包括在多播路由器公告中。如果路由器不支持 IP 多播，它就是缺省值。

multicast

指定多播路由器公告中只应包括指定的地址。如果系统不支持 IP 多播，则将不包括指定的地址。如果系统支持 IP 多播，则缺省为在给定接口支持 IP 多播时在多播路由器公告中包括地址。如果给定接口不支持 IP 多播，则将在广播路由器公告中包括指定的地址。

preference *preference*

地址作为缺省路由器地址的优先级，它相对于同一个子网上的其他路由器地址。值较大的 32 位带符号补码整数表示较高的优先级。请注意，**hex 80000000** 只能指定为 **ineligible**。缺省值为 **0**。

ineligible

指定将为给定地址分配优先级 (hex 80000000)，表示它不具备作为任何主机缺省路由的资格。

当地址不应该用作缺省路由，但给定为 ICMP 重定向中的下一跃点时，它非常有用。这样，主机就可以验证给定的地址是否启动并且可用。

路由器发现客户端

如果 IP 多播可用并已启用，或者位于接口广播地址上，则主机将通过所有主机多播地址 (**224.0.0.1**) 来监听路由器公告。在启动或重新配置时，主机可能会将多个路由器请求发送到所有路由器多播地址 **224.0.0.2** 或接口广播地址。

当收到生命周期不为零的路由器公告时，主机会将一个缺省路由安装到每个公告的地址。如果优先级不具备资格 (**ineligible**) 或地址不在连接的接口上，路由将标记为无法使用，但是将保留下来。如果优先级可用，则度量单位将设置为优先级的函数，以使用具有最佳优先级的路由。如果收到具有相同优先级的多个地址，则将使用具有最小 IP 地址的地址。这些缺省路由不可导出至其他协议。

当收到生命周期为零的路由器公告时，主机将删除包含从该路由器得到的下一跃点地址的所有路由。此外，还将删除从指向这些地址的 ICMP 重定向得到的任何路由器。如果在生命周期过期之前没有收到路由器公告来刷新这些路由，则将出现同样的情况。

路由器发现客户端语句

routerdiscovery client yes | no | on | off [{
  traceoptions *trace_options* **;**
  preference *preference* **;**
  interface *interface_list*
    [**enable**] | [**disable**]
    [**broadcast**] | [**multicast**]
    [**quiet**] | [**solicit**]
    ;
}] ;

traceoptions *trace_options*

指定 OSPF 的跟踪选项（请参阅下面的跟踪语句和 OSPF 特定跟踪选项）。

preference *preference* **;**

指定所有路由器发现缺省路由的优先级。缺省值为 **55**。

interface *interface_list*

指定适用于物理接口的参数。请注意，在惯例上与 GateD 的其他部分略有不同， **interface** 仅指定物理接口（如 **le0**、**ef0** 和 **en1**）。路由器发现客户端不具有仅适用于接口地址的参数。

enable 指定应该在指定接口上执行路由器发现。这是缺省操作。

disable 指定不应在指定接口上执行路由器发现。

broadcast 指定应该在指定接口上广播路由器请求。如果 IP 多播支持在该主机或接口上不可用，则它为缺省操作。

multicast 指定应该在指定接口上多播路由器请求。如果 IP 多播在该主机和接口上不可用，则不执行请求。缺省为在主机和接口支持时多播路由器请求。否则，将广播路由器请求。

quiet 指定即使将执行路由器发现，仍不在该接口上发送路由器请求。

solicit 指定将在该接口上发送初始路由器请求。这是缺省操作。

跟踪选项

路由器发现客户端和服务器支持 **state** 跟踪标志，它跟踪各种协议实例。

state 状态转换

路由器发现客户端和服务器不直接支持任何数据包跟踪选项，路由器发现数据包的跟踪通过 ICMP 语句来启用。

内核语句

虽然内核接口从技术上来说不是路由协议，但它具有路由协议的许多特性，GateD 将以类似于处理路由协议的方式来处理它。 GateD 选择来安装到内核转发表中的路由是内核实际将用于转发数据包的路由。

GateD 更新典型的内核转发表时必须使用的添加、删除和更改操作会耗用相当长的时间。这对于较早的路由协议（RIP、 EGP）不是问题，因为这些协议对时间要求并不特别严格，任何情况下均不会轻易地处理大量的路由。较新的协议（OSPF、BGP）具有更加严格的时间要求，通常用于处理较多的路由。当使用这些协议时，内核接口的速度就变得非常关键。

为了防止 GateD 锁定特别长的时间来安装大量路由（在实际网络中已观察到长达一分钟甚至更长），现在已经成批处理这些路由。这些批次的大小可以用下面所述的微调参数来控制，但是缺省参数通常会提供合适的功能。

在常规的关机处理中，GateD 通常会删除它安装在内核转发表中的所有路由，但带有 **retain** 标记的路由除外。或者，GateD 也可以通过不删除任何路由来保留内核转发表中的所有路由。这种情况下，将做出相应的更改，以确保带有 **retain** 标志的路由安装在表中。这在包含大量路由的系统上非常有用，因为它可以避免在 GateD 重新启动时重新安装路由。这可以大大减少从重新启动恢复所需的时间。

转发表和路由表

内核中控制数据包转发的表为转发表，在 ISO 中也称作转发信息库或 *FIB*。GateD 在内部用来存储从路由协议得到的路由信息的表为路由表，在 ISO 中称作路由信息库或 *RIB*。路由表用于收集和存储来自各种协议的路由。对于网络和掩码的每一种唯一组合，将选择活动路由，该路由将是具有最佳（数字最小）的优先级的路由。所有活动路由均安装在内核转发表中。该表中的条目是内核实际用来转发数据包的路由。

更新转发表

主要有两种方法来更新内核 *FIB*： **ioctl()** 接口和路由套接字接口。下面介绍它们的各种特性。

用 **ioctl** 接口更新转发表

转发表的 **ioctl** 接口在 *BSD 4.3* 中进行了介绍，并且在 *BSD 4.3* 中广泛分发。这是单向接口，它只允许 GateD 更新内核转发表。它具有其他一些限制：

Fixed subnet masks
: BSD 4.3 网络代码假定给定网络的所有子网均具有相同的子网掩码。该限制是由内核强加的。网络掩码不会存储在内核转发表中，但将在转发数据包时通过在同一个网络上搜索接口确定。

One way interface
: GateD 能够更新内核转发表，但是它无法注意到转发表的其他修改。GateD 能够监听 ICMP 消息并猜测内核如何用 ICMP 重定向响应来更新转发表。

Blind updates
: GateD 无法检测到因为系统管理员使用 *route* 命令而造成的转发表更改。运行 GateD 时，强烈建议不要在使用 **ioctl()** 接口的系统上使用 *route* 命令。

Changes not supported
: 在所有已知的实现中，均不支持更改存在于内核中的路由，否则必须删除该路由并添加新路由。

用路由套接字接口更新转发表

内核转发表的路由套接字接口在 *BSD 4.3 Reno* 中进行了介绍，在 *BSD 4.3 Net/2* 中广泛分发，并在 *BSD 4.4* 中进行了改进。该接口只是内核和 GateD 用来交换消息的套接字，与 UDP 套接字类似。与 **ioctl()** 接口相比，它具有以下几项优点：

Variable subnet masks
: 网络掩码显式传递到内核。这样，可以在同一个网络的子网上使用不同的掩码。它还允许使用其掩码比自然掩码更为通用的路由。这种路由称作无类路由。

Two way interface
: 不仅 GateD 能够用该接口更改内核转发表，而且内核也可向 GateD 报告对转发表的更改。最有趣的是重定向已修改内核转发表的指示；这意味着 GateD 不再需要监视 ICMP 消息来了解重定向。此外，还将指示内核是否处理重定向，GateD 是否可以安全地忽略内核未处理的消息。

Updates visible
: 其他进程（包括 *route* 命令）对路由表的更改通过路由套接字来接收。这样， GateD 可确保内核转发表与路由表同步。此外，系统管理员还能够在 GateD 运行过程中用 *route* 命令执行一些操作。

Changes supported
: 活动的 *change* 消息允许以不可分割的方式更改内核中的路由。路由套接字代码的某些早期版本在 change 消息处理方面存在缺陷。编译时间和配置时间选项可能导致使用删除和添加序列来代替 change 消息。

Expandable

通过添加新的消息类型，可以添加新的内核/GateD 通信级别。

读取转发表

GateD 启动时，将读取内核转发表，并将相应的路由安装在路由表中。这些路由称作遗留路由，它们将在配置的时间间隔（缺省为 3 分钟）之后超时，或者在得到更重要的路由时立即超时。这样，转发就会在路由协议开始获取路由的期间发生。

主要有三种方法来从内核中读取转发表。

通过 **kmem** 读取转发表

在许多系统（尤其是在基于 BSD 4.3 的系统）上，GateD 必须了解内核数据结构，它可以进入内核读取转发表的当前状态。该方法比较缓慢，如果在 GateD 读取内核转发表的过程中将其更新，则可能出错。如果系统管理员使用 *route* 命令，或者在 GateD 启动期间收到 ICMP 重定向消息，则可能出现这种情况。

由于疏忽，某些基于 BSD 4.3 Reno 或更高版本的系统（如 OSF/1）不具有下面所述的 **getkerninfo()** 系统调用，通过该系统调用，GateD 无需知道内核内部结构即可从内核中读取路由。在这些系统上，必须通过在内核内存中查找来读取内核中的内核基数树。这甚至比读取基于散列的转发表更容易出错。

通过 **getkerninfo/sysctl** 读取转发表

除了路由套接字之外，BSD 4.3 Reno 还引入了 **getkerninfo()** 系统调用。通过该调用，用户进程（如 GateD）无需了解内核数据结构即可读取内核中的各种信息。对于转发表，它将以不可分割的方式作为一系列路由套接字消息返回到 GateD。这将防止在 GateD 读取转发表时出现与转发表更改相关的问题。

BSD 4.4 将 **getkerninfo()** 接口更改为 **sysctl()** 接口，后者采用不同的参数，但在其他方面完全相同。

通过 **OS** 特定方法读取转发表

某些操作系统（如 SunOS 5）会定义其自己的内核转发表读取方法。SunOS 5 版本在概念上类似于 **getkerninfo()** 方法。

读取接口列表

GateD 的内核支持子系统负责定期读取内核物理和协议接口的状态。 GateD 将检测接口列表中的更改并通知协议，使其能够启动或停止实例或对等端。接口列表以下列两种方式之一来读取：

用 **SIOCGIFCONF** 读取接口列表

在基于 BSD 4.3、4.3 Reno 和 4.3 Net/2 的系统上， **SIOCGIFCONF ioctl** 接口用于读取内核接口列表。如果使用该方法， **SIOCGIFCONF** 调用将返回接口列表以及有关这些接口的一些基本信息。要得到其他信息，必须通过发出其他 ioctl 来获取接口网络掩码、标志、MTU、度量单位、目标地址（对于点对点接口）和广播地址（对于支持广播的接口）。

GateD 每隔 15 秒钟重新读取一次该列表以查找更改。如果路由套接字正在使用，也会在每次收到指示路由配置更改的消息时重新读取。如果收到 **SIGUSR2** 信号，也将导致 GateD 重新读取该列表。该时间间隔可以在接口配置中显式地配置。

用 **sysctl** 读取接口列表

BSD 4.4 新增了通过 *sysctl* 系统调用读取内核接口列表的功能。接口状态以不可分割的方式作为 GateD 为获取必需信息而分析的路由套接字消息的列表返回。

BSD 4.4 还添加了路由套接字消息，以立即报告接口状态更改。这样，GateD 就可以快速地响应接口配置中的更改。

使用该方法时，GateD 仅每隔一分钟就重新读取一次接口列表。它也会在出现路由表更改指示以及收到 **SIGUSR2** 时重新读取。该时间间隔可以在接口配置中显式地配置。

读取接口物理地址

getkerninfo() 和 **sysctl()** 的后期版本将接口物理地址作为接口信息的一部分返回。在不返回该信息的大多数系统上，GateD 将扫描内核物理接口列表，为设置 **IFF_BROADCAST** 的接口查找该信息，并假定其驱动程序的处理方式与以太网驱动程序的处理方式相同。在某些系统（如 *SunOS 4* 和 *SunOS 5*）上，将使用系统特定接口来获取该信息。

接口物理地址对于 IS-IS 和 IP 协议非常有用，它们当前未使用，但可能会在将来使用。

读取内核变量

启动时，GateD 会从内核中读取一些特殊变量。这通常使用 *nlist*（或 *kvm_nlist*）系统调用来完成，但某些系统会使用不同的方法。

变量读取的内容包括 *UDP* 校验和创建和生成的状态、*IP* 转发以及内核版本（为了提供信息）。在直接从内核内存中读取路由表的系统上，将读取哈希表或基数树路由表的根。在接口物理地址不通过其他方式提供的系统上，将读取接口列表的根。

特殊路由标志

基于后期 *BSD* 的内核支持本文所述的特殊路由标志。

RTF_REJECT

带有 **RTF_REJECT** 标志的路由将导致丢弃数据包并将 **unreachable** 消息发送到数据包初始发送方，而不是像常规路由一样转发数据包。该标志仅在指向环回接口的路由上才有效。

RTF_BLACKHOLE

与 **RTF_REJECT** 标志类似，带有 **RTF_BLACKHOLE** 标志的路由将导致丢弃数据包，但不发送 **unreachable** 消息。该标志仅在指向环回接口的路由上才有效。

RTF_STATIC

GateD 启动时，将读取内核转发表中的所有当前路由。除了接口路由之外，它通常将其他所有路由标记为上次 GateD 运行的遗留路由，并在几分钟后将其删除。这意味着用 *route* 命令添加的路由在 GateD 启动后将不会保留。

为了解决该问题，添加了 **RTF_STATIC** 标志。当使用 *route* 命令来安装除接口路由之外的路由时，该命令将设置 **RTF_STATIC** 标志。该标志向 GateD 表示该路由是由系统管理员添加的，应该保留。

内核配置

kernel {
  options
    [**nochange**]
    [**noflushatexit**]
    [**remnantholdtime** *time*]
    ;
  routes *number* **;**
  flash
    [**limit** *number*]
    [**type interface** | **interior** | **all**]
    ;
  background
    [**limit** *number*]
    [**priority flash** | **higher** | **lower**]
    ;
  traceoptions *trace_options* **;**
} ;

options *option_list*

配置内核选项。有效选项包括:

nochange

在支持路由数据包的系统上，它确保不执行更改操作，而仅执行删除和添加操作。对于无法执行更改操作的早期路由套接字代码版本，它非常有用。

noflushatexit

在常规的关机处理过程中，GateD 将从内核转发表中删除所有不带 **retain** 指示的路由。 **noflushatexit** 选项可防止在关机时删除路由。它将更改并添加路由，以确保所有带 **retain** 标记的路由均已安装。

这在包含上千条路由的系统上非常有用。启动时， GateD 将通知位于内核转发表中的路由，而不是再次添加这些路由。

remnantholdtime *time*

通常，启动时从内核转发表中读取的遗留路由将在三分钟后超时，或者在被覆盖时立即超时。通过该选项，可以将时间间隔配置为介于 0 和 15 分钟之间的值。如果将其设置为零，则将立即删除这些路由。

routes *number*

在某些系统上，内核内存资源非常珍贵。通过该参数，可以对 GateD 在内核中安装的最大路由数设置限制。通常， GateD 按照接口/内部/外部的顺序添加/更改/删除路由。它会将接口路由排在

队列最前面，接着是内部路由，最后是外部路由，然后再从头处理队列。如果指定该参数并达到限制，则 GateD 将改为扫描该列表两次。在首次扫描时，它将执行删除，并且还删除所有更改的路由，将列队的的更改变为添加。它将重新扫描列表，按接口/内部/外部的顺序执行添加，直到它再次达到限制。相对于外部路由，它将倾向于首选内部路由。缺省值是限制内核转发表中的路由数。

flash 当路由变化时，通知协议的进程称作瞬时更新。首先通知的是内核转发表接口。一次瞬时更新通常可以处理最多 20 个接口路由。 **flash** 命令允许微调这些参数。

limit *number*

指定一次瞬时更新可处理的最大路由数。缺省值是 **20**。值 **-1** 将导致在瞬时更新期间处理指定类型的所有未决路由更改。

type interface | interior | all

指定将在瞬时更新期间处理的路由类型。 **Interior** 指定还将安装内部路由（请参阅内部网关协议的定义）。 **All** 指定同时包括外部协议（请参阅外部网关协议的定义）。缺省值是 **interface**，它指定瞬时更新期间仅安装接口路由。

如果指定 **flash limit -1 all**，则 瞬时更新期间将安装所有路由；这可模拟早期版本的 GateD 的行为。

background

由于瞬时更新期间通常仅安装接口路由，剩余路由将在后台成批处理（即，在没有接收路由协议通信量时）。通常，一次将安装 120 条路由以允许执行其他任务，该后台处理的优先级低于瞬时更新。通过以下参数可微调相应参数：

limit *number*

指定一个批次可处理的路由数。缺省值是 120。

priority flash | higher | lower

指定内核更新批处理相对于瞬时更新处理的优先级。缺省值是 **lower**，这表示首先处理瞬时更新。要以相同的优先级处理内核更新和瞬时更新，请指定 **flash**；要以较低的优先级处理内核更新，请使用 **lower**。

跟踪选项

虽然内核接口从技术上来说不是路由协议，但许多情况下会将其当作路由协议处理。以下两个符号在命令行上输入时将有效，因为使用它们的代码将在分析跟踪文件之前执行。

symbols

nlist() 或类似的接口从内核中读取的符号。

iflist 接口列表扫描。由于第一次接口列表扫描在分析配置文件之前执行，因此该选项在命令行上输入时才有用。

以下跟踪选项只能在配置文件中指定。它们在命令行上无效。

remnants

在 GateD 启动时从内核读取的路由。

request

GateD 发出的内核转发表路由添加/删除/更改请求。

静态语句

Static 语句定义 GateD 使用的静态路由。单个 **static** 语句可以指定任意数量的路由。 **static** 语句出现在 gated.conf 中的协议语句之后和控制语句之前。可以指定任意数量的 **static** 语句，每个语句可包含任意数量的静态路由定义。这些路由可以由具有更佳优先级值的路由覆盖。

static {
  (**host** *host*) | **default** |
  (*network* [(**mask** *mask*) | (**masklen** *number*)])
    gateway *gateway_list*
    [**interface** *interface_list*]
    [**preference** *preference*]
    [**retain**]
    [**reject**]
    [**blackhole**]
    [**noinstall**] **;**
  (*network* [(**mask** *mask*) | (**masklen** *number*)])
    interface *interface*
    [**preference** *preference*]
    [**retain**]
    [**reject**]
    [**blackhole**]
    [**noinstall**] **;**
} ;

host *host* **gateway** *gateway_list*
(*network* [(**mask** *mask*) | (**masklen** *number*)])
default gateway *gateway_list*

这是 static 语句最常用的格式。它定义通过一个或多个网关的静态路由。当列出的一个或多个 **gateways** 在直接连接接口上可用时，将安装静态路由。如果多个合格的网关可用，它们将受到支持的多路径目标数的限制（在 Unix 上，该编译时参数目前几乎始终是一）。

静态路由的参数包括：

interface *interface_list*

如果指定该参数，只有当网关位于其中一个接口上时，才认为它们有效。有关 *interface_list* 的说明，请参阅接口列表说明部分。

preference *preference*

该选项选择该静态路由的优先级。优先级控制该路由如何与其他协议的路由进行竞争。缺省的优先级是 60。

retain 通常，GateD 会在正常关机过程中删除内核转发表中除接口路由之外的所有路由。**retain** 选项可用于防止删除特定的静态路由。它可用于确保某种路由在 GateD 未运行时可用。

reject 带有 **reject** 标志的路由将导致丢弃数据包并将 **unreachable** 消息发送到数据包初始发送方，而不是像常规路由一样转发数据包。如果指定该选项，将使该路由安装为 reject 路由。并非所有内核转发引擎都支持 reject 路由。

blackhole

除了不支持 **unreachable** 消息之外，**blackhole** 路由与 **reject** 路由基本相同。

noinstall

通常，具有最低优先级的路由将安装在内核转发表中，它是导出至其他协议的路由。在路由上指定 **noinstall** 时，该路由将不会在活动时安装到内核转发表中，但它仍具备导出至其他协议的资格。

(*network* [(**mask** *mask*) | (**masklen** *number*)]) **interface** *interface*

该格式定义用于为一个接口上的多个网络地址提供原始支持的静态接口路由。 **preference**、**retain**、 **reject**、**blackhole** 和 **noinstall** 选项与上述选项相同。

控制语句概述

控制语句控制从路由对等端导入的路由以及导出至这些对等端的路由。它们是最后包括在 gated.conf 文件中的语句。控制语句包括：

- Import（引入）语句
- Export（导出）语句
- Aggregate（汇聚）语句
- Generate（生成）语句

路由过滤

路由通过指定配置语言来过滤，配置语言将按目标或者按目标和掩码来匹配特定的路由组。路由过滤器在 **martians**、**import**、**export** 语句以及其他一些位置中使用。

在没有找到匹配项时所执行的操作取决于环境，例如 **import** 和 **export** 路由过滤器在列表末尾假定 **all reject ;**。

路由将匹配适用的最具体的过滤器。如果指定多个具有相同目标、掩码和修饰符的过滤器，则将生成错误。

过滤语法

network [**exact** | **refines**]

network **mask** *mask* [**exact** | **refines**]

network **masklen** *number* [**exact** | **refines**]

all

default

host *host*

它们是路由过滤器的所有可能格式。并非所有这些格式均在所有位置可用，例如 **host** 和 **default** 格式对于 **martians** 无效。

大多数情况下，可以指定与过滤器的环境相关的附加参数。例如，在 **martian** 语句上，可以指定 **allow** 关键字，在 **import** 语句上，可以指定优先级，在 **export** 上，可以指定度量单位。

network [**exact** | **refines**]

network **mask** *mask* [**exact** | **refines**]

network **masklen** *number* [**exact** | **refines**]

虽然掩码是以下列简写形式包含的，但是匹配通常需要地址和掩码。这三种格式在指定掩码的方式上有所不同。在第一种格式中，包含的掩码为网络的自然掩码。在第二种格式中，掩码将显式指定。在第三种格式中，掩码由连续 1 位的个数指定。

如果没有指定任何附加参数，则将匹配处于网络给定范围的任何目标。将忽略目标的掩码。如果指定自然网络，则网络、任何子网和任何主机将匹配。通过两个可选的修饰符，也可考虑目标的掩码:

exact 该参数指定目标的掩码必须完全匹配提供的掩码。它用于匹配网络，而不是该网络的任何子网或主机。

refines 指定目标的掩码必须比过滤器掩码更具体（更长）。它用于匹配网络的子网和（或）主机，但不是网络。

all 该条目匹配任何内容。它等效于:

0.0.0.0 mask 0.0.0.0

default 匹配 **default** 路由。要进行匹配，地址必须是缺省地址，而掩码必须全为零。它等效于:

0.0.0.0 mask 0.0.0.0 exact

host *host*

匹配特定的主机。要进行匹配，地址必须完全匹配指定的主机，网络掩码必须是主机掩码（均为 1）。它等效于:

host **mask 255.255.255 exact**

匹配 **AS** 路径

AS 路径是路由信息在到达该路由器之前通过的自治系统的列表以及 AS 路径来源的指示符。该信息可用于首选目标网络的特定路径。使用 GateD 完成该任务的主要方法是指定在导入和导出路由时应用于 AS 路径的模式列表。

路由通过的每个自治系统会将其 AS 编号添加到 AS 路径的开头。

来源信息将详述 AS 路径信息的完整性。**igp** 的来源表示路由是从内部路由协议得到的，并且很可能是完整的。**egp** 的来源表示路由是从不支持 AS 路径的外部路由协议（如 EGP）得到的，并且路径很可能是不完整的。当确定路径信息不完整时，将使用 **incomplete** 的来源。

AS 路径正则表达式在 RFC 1164 的第 4.2 节定义。

AS 路径匹配语法

AS 路径使用以下语法进行匹配。

aspath *aspath_regexp* **origin any** | ([**igp**] [**egp**] [**incomplete**])

它指定匹配与具有指定来源的 *aspath_regexp* 匹配的 AS。

AS 路径正则表达式

从技术上看，AS 路径正则表达式是以字母表作为 AS 编号集的正则表达式。AS 路径正则表达式由一个或多个 AS 路径表达式组成。AS 路径表达式由 AS 路径条件和 AS 路径运算符组成。

AS 路径条件

AS 路径条件是以下三个对象之一:

autonomous_system
.
(*aspath_regexp* **)**

其中

autonomous_system	任何有效的自治系统编号，从 1 到 65534（包括这两个数字）。
.	匹配任何自治系统编号。
(*aspath_regexp* **)**	为组子表达式（运算符）添加括号如 * 或 **?** 处理单个元素或括号中的正则表达式。

AS 路径运算符

AS 路径运算符是以下对象之一:

aspath_term **{m,n}**
aspath_term **{m}**
aspath_term **{m,}**
aspath_term *****
aspath_term **+**
aspath_term **?**
aspath_term | *aspath_term*

aspath_term **{m,n}**

后接 **{m,n}**（其中 m 和 n 都是非负整数，且 m <= n）的正则表达式表示至少 *m* 次且最多 *n* 次重复。

aspath_term **{m}**
后接 **{m}**（其中 m 是正整数）的正则表达式表示正好 *m* 次重复。

aspath_term **{m,}**
后接 **{m,}**（其中 m 是正整数）的正则表达式表示 m 次或更多次重复。

aspath_term *****
后接 ***** 的 AS 路径条件表示零次或更多次重复。它是 **{0,}** 的简写形式。

aspath_term **+**
后接 **+** 的正则表达式表示一次或更多次重复。它是 **{1,}** 的简写形式。

aspath_term **?**
后接 **?** 的正则表达式表示零次或一次重复。它是 **{0,1}** 的简写形式。

aspath_term **|** *aspath_term*
匹配左侧的 AS 条件或右侧的 AS 条件。

Import（引入）语句

从路由协议导入路由和在 GateD 路由数据库中安装路由由 **import** 语句控制。根据源协议的不同，**import** 语句的格式将有所不同。

指定优先级

在所有情况下，必须指定两个关键字之一来控制路由如何与其他协议竞争:

restrict
preference *preference*

restrict 指定路由表中不需要的路由。某些情况下，这意味着不将路由安装在路由表中。其他情况下，这意味着使用负的优先级安装路由；这将防止其进入活动状态，因此它们将不会安装在转发表中，也不会导出至其他协议。

preference *preference*
指定在比较该路由和来自其他协议的其他路由时所使用的优先级值。在任意给定路由上具有可用的最低优先级的路由将成为活动路由，安装在转发表中，并且具备导出至其他协议的资格。缺省优先级通过各个协议来配置。

路由过滤器

所有格式均允许下面所示的路由过滤器。有关其工作机制的详细解释，请参阅有关路由过滤器的部分。如果未指定路由过滤（当 **restrict** 在语句的第一行上指定时），来自指定源的所有路由将匹配该语句。如果指定任意过滤器，则将仅导入匹配指定过滤器的路由。换而言之，如果指定任何过滤器，则在列表的末尾假定 **all restrict;**。

network [**exact** | **refines**]
network **mask** *mask* [**exact** | **refines**]

network **masklen** *number* [**exact** | **refines**]

default

host *host*

从 BGP 和 EGP 导入路由

import proto bgp | **egp autonomoussystem** *autonomous_system*
  restrict ;
import proto bgp | **egp autonomoussystem** *autonomous_system*
  [**preference** *preference*] **{**
  route_filter [**restrict** | (**preference** *preference*)] **;**
} ;

import proto bgp aspath *aspath_regexp*
  origin any | ([**igp**] [**egp**] [**incomplete**])
  restrict ;
import proto bgp aspath *aspath_regexp*
  origin any | ([**igp**] [**egp**] [**incomplete**])
  [**preference** *preference*] **{**
  route_filter [**restrict** | (**preference** *preference*)] **;**
} ;

EGP 导入可以由自治系统来控制。 BGP 还支持使用 AS 路径正则表达式控制传播，这将在“匹配 AS 路径”一节中介绍。请注意，EGP 和 BGP 版本 2 和 3 仅支持自然网络传播，因此 **host** 和 **default** 路由过滤器是毫无意义的。BGP 版本 4 支持随连续网络掩码传播任何目标。

EGP 和 BGP 均存储任何因为未在路由过滤器中提及而隐式拒绝的路由，或者在路由表中使用 **restrict** 关键字以负优先级显式拒绝的路由。负优先级防止路由进入活动状态，这会防止将该路由安装在转发表中，或者导出至其他协议。在更改导入策略的情况下，这将减少在重新配置时断开和重新建立会话的必要。

从 RIP、HELLO 和重定向导入路由

import proto rip | **hello** | **redirect**
  [(**interface** *interface_list*) | (**gateway** *gateway_list*)]
  restrict ;
import proto rip | **hello** | **redirect**
  [(**interface** *interface_list*) | (**gateway** *gateway_list*)]
  [**preference** *preference*] **{**
  route_filter [**restrict** | (**preference** *preference*)] **;**
} ;

RIP、HELLO 和重定向路由的导入可以通过协议、源接口和源网关中的任意一种来控制。如果指定了多种，则从最一般（协议）到最具体（网关）的顺序对其进行处理。

RIP 和 HELLO 不支持使用优先级在相同协议的路由之间选择。这由协议度量单位来处理。由于拒绝的路由具有

很短的更新时间间隔，这些协议不保存拒绝的路由。

从 **OSPF** 导入路由

import proto ospfase [**tag** *ospf_tag*] **restrict ;**

import proto ospfase [**tag** *ospf_tag*]

[**preference** *preference*] **{**

route_filter [**restrict** | (**preference** *preference*)] **;**

} ;

由于 OSPF 的特性，只能控制 ASE 路由的导入。 OSPF 区域内和区域间路由始终以优先级 10 导入到 GateD 路由表。如果指定了标志，则 import 子句将仅应用于带有指定标志的路由。

只有在用作 AS 边界路由器时，才能限制 OSPF ASE 路由的导入。这通过指定 **export ospfase** 子句来完成。通过指定空的 export 子句，可以限制在未导出 ASE 时的 ASE 导入。

与其他内部协议相似，优先级可用于在 OSPF ASE 路由之间进行选择，这通过 OSPF 成本来完成。策略拒绝的路由以负优先级存储在表中。

Export（导出）语句

import 语句控制从 GateD 所用的其他系统上接收了哪些路由，**export** 语句控制 GateD 将哪些路由广播到其他系统。与 **import** 语句类似，各个协议的 **export** 语句的语法会略有不同。 **export** 语句的语法与 **import** 语句的语法类似，它们的许多参数均具有相同的含义。它们之间的主要区别在于路由导入仅由源信息来控制，而路由导出由目标和源来控制。

给定 **export** 语句的外层部分指定所控制的路由信息的目标。中间部分限制您希望考虑的输入源。最内层的部分是用于选择单个路由的路由过滤器。

指定度量单位

度量单位最具体的规范是应用于所导出的路由的规范。可以为度量单位指定的值取决于该 Export（导出）语句引用的目标协议。

restrict

metric *metric*

restrict 指定不应导出任何内容。如果在 **export** 语句的目标部分执行，则指定不应将任何内容导出至该目标。如果在源部分指定，则指定不应将来自指定源的任何内容导出至该目标。如果指定为路由过滤器的一部分，则指定不应导出匹配该过滤器的路由。

metric *metric*

指定在导出至指定目标时使用的度量单位。

路由过滤器

所有格式均允许下面所示的路由过滤器。有关其工作机制的详细解释，请参阅有关路由过滤器的部分。如果未指定路由过滤（当 **restrict** 在语句的第一行上指定时），来自指定源的所有路由将匹配该语句。如果指定任意过滤器，则将仅导出匹配指定过滤器的路由。换而言之，如果指定任何过滤器，则在列表的末尾假定 **all restrict ;**。

network [**exact** | **refines**]
network **mask** *mask* [**exact** | **refines**]
network **masklen** *number* [**exact** | **refines**]
default
host *host*

指定目标

如上所述，根据所应用的协议，**export** 语句的语法将有所不同。在所有情况下均适用的是度量单位的指定。所有协议均定义一个缺省度量单位用于导出的路由，大多数情况下它可以在 Export（导出）语句的多个级别上覆盖。

下面介绍如何指定所导出的路由信息的源 (export_list)。

导出至 **EGP** 和 **BGP**

export proto bgp | **egp as** *autonomous system*
  restrict ;
export proto bgp | **egp as** *autonomous system*
  [**metric** *metric*] **{**
  export_list **;**
} ;

向 EGP 和 BGP 的导出可以通过自治系统进行控制，相同的策略将应用于 AS 中的所有路由器。 EGP 度量单位的范围是从 0 到 255（包括这两个数字），其中 0 是最优值。

BGP 度量单位是 16 位无符号数字。其范围是从 0 到 65535（包括这两个数字），其中 0 是最优值。由于 BGP 版本 4 实际上支持 32 位无符号数字，因此 GateD 尚不支持。

如果未指定任何导出策略，则将仅导出连接接口的路由。如果指定了任何策略，则将覆盖缺省值；必须显式指定应该导出的所有内容。

请注意，EGP 和 BGP 版本 2 和 3 仅支持自然网络传播，因此 **host** 和 **default** 路由过滤器是毫无意义的。BGP 版本 4 支持随连续网络掩码传播任何目标。

导出至 **RIP** 和 **HELLO**

export proto rip | **hello**
  [(**interface** *interface_list*) | (**gateway** *gateway_list*)]
  restrict ;
export proto rip | **hello**
  [(**interface** *interface_list*) | (**gateway** *gateway_list*)]
  [**metric** *metric*] **{**
  export_list **;**
} ;

向 RIP 和 HELLO 的导出通过协议、接口或网关中的任何一种来控制。如果指定了多种，则从最一般（协议）到最具体（网关）的顺序对其进行处理。

无法为将 RIP 路由导出至 RIP 或者将 HELLO 路由导出至 HELLO 设置度量单位。这种尝试将以无提示方式忽略。

如果未指定任何导出策略，RIP 和接口路由将导出至 RIP，HELLO 和接口路由将导出至 HELLO。如果指定了任何策略，则将覆盖缺省值。必须显式指定应该导出的任何内容。

RIP 版本 1 和 HELLO 假定共享网络的所有子网均具有相同的子网掩码，因此它们只能传播该网络的子网。RIP 版本 2 取消了该限制，能够在不发送兼容版本 1 的更新时传播所有路由。

要通过 RIP 或 HELLO 声明指定环回接口下一跃点的路由（静态或内部生成的缺省路由），必须在 **export** 子句中的某一级别指定度量单位。仅为 RIP 或 HELLO 设置缺省度量单位是不够的。这是一种安全措施，可验证该声明是计划之内的声明。

导出至 **OSPF**

export proto ospfase [**type 1** | **2**] [**tag** *ospf_tag*]
  restrict ;
export proto osfpase [**type 1** | **2**] [**tag** *ospf_tag*]
  [**metric** *metric*] **{**
  export_list **;**
} ;

无法通过将路由从 GateD 路由表导出至 OSPF 来创建 OSPF 区域内或区域间路由。只能从 GateD 路由表导出至 OSPF ASE 路由。此外，无法控制 OSPF 路由在 OSPF 协议内的传播。

存在两种 OSPF ASE 路由：类型 *1* 和类型 *2*，有关这两种类型的详细解释，请参阅 OSPF 协议配置。缺省类型由 **ospf** 子句的 **defaults** 次子句定义。这可以通过 **export** 语句上的指定来覆盖。

OSPF ASE 路由还具有传送标志的规定。它是任意 32 位数字，可以用来在 OSPF 路由器上过滤路由信息。有关 OSPF 标志的详细信息，请参阅 OSPF 协议配置。 **ospf defaults** 子句指定的缺省标志可以由 **export** 语句上指定的标志覆盖。

指定源

导出列表根据路由的来源指定导出，其语法根据具体的源而有所不同。

导出 **BGP** 和 **EGP** 路由

proto bgp | **egp autonomoussystem** *autonomous_system*
  restrict ;
proto bgp | **egp autonomoussystem** *autonomous_system*
  [**metric** *metric*] **{**
  route_filter [**restrict** | (**metric** *metric*)] **;**
} ;

源自治系统可以指定 BGP 和 EGP 路由。所有路由均可按 AS 路径导出，有关详细信息，请参阅下文。

导出 **RIP** 和 **HELLO** 路由

proto rip | **hello**
[(**interface** *interface_list*) | (**gateway** *gateway_list*)]
restrict ;
proto rip | **hello**
[(**interface** *interface_list*) | (**gateway** *gateway_list*)]
[**metric** *metric*] **{**
route_filter [**restrict** | (**metric** *metric*)] **;**
} ;

RIP 和 HELLO 路由可以按协议、源接口和（或）源网关导出。

导出 **OSPF** 路由

proto ospf | **ospfase restrict ;**
proto ospf | **ospfase** [**metric** *metric*] **{**
route_filter [**restrict** | (**metric** *metric*)] **;**
} ;

OSPF 和 OSPF ASE 路由可以导出至其他协议。有关按标志导出的信息，请参阅下文。

从非路由协议导出路由
通过接口的非路由

proto direct | **static** | **kernel**
[(**interface** *interface_list*)]
restrict ;
proto direct | **static** | **kernel**
[(**interface** *interface_list*)]
[**metric** *metric*] **{**
route_filter [**restrict** | (**metric** *metric*)] **;**
} ;

这些协议可以按协议或者按下一跃点的接口导出。这些协议包括:

direct 直接连接接口的路由。

static 在 **static** 子句中指定的静态路由。

kernel 在包含路由套接字的系统上，从路由套接字得到的路由以内核的协议安装在 GateD 路由表中。这些路由可以通过引用该协议来导出。当需要具有包含 **route** 命令的脚本安装路由并将其传播到其他路由协议时，这非常有用。

按协议的非路由

proto default | **aggregate**
restrict ;

proto default | **aggregate**
[**metric** *metric*] **{**
route_filter [**restrict** | (**metric** *metric*)] **;**
} ;

这些协议只能按协议引用。

default 引用 **gendefault** 选项创建的路由。建议改用路由生成。

aggregate
在使用 **aggregate** 和 **generate** 语句时引用根据其他路由进行同步的路由。有关详细信息，请参阅“路由汇聚”一节。

按 AS 路径导出

proto *proto* | **all aspath** *aspath_regexp*
origin any | ([**igp**] [**egp**] [**incomplete**])
restrict ;
proto *proto* | **all aspath** *aspath_regexp*
origin any | ([**igp**] [**egp**] [**incomplete**])
[**metric** *metric*] **{**
route_filter [**restrict** | (**metric** *metric*)] **;**
} ;

如果 BGP 已配置，则会在将路由添加到路由表时给所有路由分配 AS 路径。对于所有内部路由，该 AS 路径将 IGP 指定为来源，且 AS 路径中不存在 AS（导出路由时将添加当前 AS）。对于 EGP 路由，该 AS 路径将 EGP 指定为来源，将源 AS 指定为路径。对于 BGP 路由， AS 路径按照从 BGP 获取的方式存储。

AS 路径正则表达式在“匹配 AS 路径”一节中介绍。

按路由标志导出

proto *proto* | **all tag** *tag* **restrict ;**
proto *proto* | **all tag** *tag*
[**metric** *metric*] **{**
route_filter [**restrict** | (**metric** *metric*)] **;**
} ;

OSPF 和 RIP 版本 2 当前均支持标志，所有其他协议始终具有标志零。可以根据该标志选择导出路由的源。如果路由在导出至给定路由协议时按标志进行分类，这将非常有用。

路由汇聚

路由汇聚是在存在特定路由的前提下生成更为通用的路由的方法。例如，如果存在通过 RIP 得到的网络的一个或多个子网，则在自治系统边界使用它生成将通过 EGP 声明的网络的路由。早期版本的 GateD 会自动执行该功能：在给定该自然网络的子网接口时生成自然网络的汇聚路由（使用早期的类 A、B 和 C 概念）。但是，这样做并非始终是正确的，随着无类域间路由的出现，这样做在更多的时候反而是错误的，因此必须显式配置汇聚。除

非在 **aggregate** 语句中显式请求，否则将不执行汇聚。

路由汇聚也由区域性和全国性网络用来减少传递的路由信息量。通过小心地将网络地址分配到客户端，区域性网络只需向区域性网络声明一条路由，而不是数百条路由。

汇聚路由并不由汇聚路由的初始发送方，而仅由接收方实际用于数据包转发（如果希望这样）。如果路由器收到与导致生成汇聚路由的组件路由之一不匹配的数据包，它应该以 ICMP 网络无法连接消息进行响应。这是为了防止未知组件路由的数据包跟随缺省路由进入另一个网络，在该网络中它们转发回边界路由器，并且不停地往返传输，直到 TTL 过期。只有在带有拒绝路由支持的系统上，才能为汇聚的缺少部分发送无法连接消息。

汇聚的细微变化是基于某些条件是否存在而生成路由。这有时称作最后的路由。该路由从以优先级值最小（优先级最高）指定的提供方继承下一跃点和 AS 路径。最常见的用法是基于对等端中的路由是否存在于相邻主干上来生成缺省值。

汇聚和生成语法

aggregate default
| (*network* [(**mask** *mask*) | (**masklen** *number*)])
[**preference** *preference*] [**brief**] **{**
proto [**all** | **direct** | **static** | **kernel** | **aggregate** | *proto*]
[(**as** *autonomous system*) | (**tag** *tag*)
| (**aspath** *aspath_regexp*)]
restrict ;
proto [**all** | **direct** | **static** | **kernel** | **aggregate** | *proto*]
[(**as** *autonomous system*) | (**tag** *tag*)
| (**aspath** *aspath_regexp*)]
[**preference** *preference*] **{**
route_filter [**restrict** | (**preference** *preference*)] **;**
} ;
} ;

generate default
| (*network* [(**mask** *mask*) | (**masklen** *number*)])
[**preference** *preference*] **{**
[(**as** *autonomous system*) | (**tag** *tag*)
| (**aspath** *aspath_regexp*)]
restrict ;
proto [**all** | **direct** | **static** | **kernel** | **aggregate** | *proto*]
[(**as** *autonomous system*) | (**tag** *tag*)
| (**aspath** *aspath_regexp*)]
[**preference** *preference*] **{**
route_filter [**restrict** | (**preference** *preference*)] **;**

```
  };
};
```

匹配路由过滤器的路由称作参与路由。它们按照所应用的汇聚优先级进行排序。如果多个参与路由具有相同的汇聚优先级，则使用路由的优先级来将路由排序。汇聚路由的优先级将是具有最低汇聚优先级的参与路由的优先级。

preference *preference*
指定要分配给所得汇聚路由的优先级。缺省的优先级是 130。

brief 用于指定应该将 AS 路径截断到最长的公共 AS 路径。缺省值是构建由所有参与 AS 路径的 SET 和 SEQUENCE 组成的 AS 路径。

proto *proto*
除了列出的特殊协议之外，可以从 GateD 支持的任何协议（并且当前配置到 GateD 中）中选择参与协议。

as *autonomous_system*
将路由选择限制在从指定自治系统得到的路由。

tag *tag* 将路由选择限制在带有指定标记的路由。

aspath *aspath_regexp*
将路由选择限制在匹配指定 AS 路径的路由。

restrict 指示不将这些路由视为指定汇聚的参与路由。指定协议可以是 GateD 所支持的任何协议。

route_filter
请参阅下文。

路由只能参与比其自身更为通用的汇聚路由；它必须匹配其掩码下的汇聚。任何给定路由只能参与一条汇聚路由（它将是所配置的最具体的路由），但是汇聚路由可以参与更具一般性的汇聚。

路由过滤器

所有格式均允许下面所示的路由过滤器。有关其工作机制的详细解释，请参阅有关路由过滤器的部分。如果未指定路由过滤（当 **restrict** 在语句的第一行上指定时），来自指定源的所有路由都将匹配该语句。如果指定任意过滤器，则只将匹配指定过滤器的路由当作参与路由。换而言之，如果指定任何过滤器，则在列表的末尾假定 **all restrict ;**。

network [**exact** | **refines**]
network **mask** *mask* [**exact** | **refines**]
network **masklen** *number* [**exact** | **refines**]
default
host *host*

术语表

下面定义本文中的描述所使用的术语:

“邻接关系”

在选定的相邻路由器之间形成的关系，用于交换路由信息。并非每一对相邻路由器都具有邻接关系。

“自治系统”

单一技术管理下的一组路由器，它们使用内部网关协议和通用度量单位在 AS 中路由数据包，并使用外部网关协议将数据包路由到其他 AS。由于这一传统定义的普及，单个 AS 通常会使用多个内部网关协议，并在有时使用 AS 中的多组度量单位。自治系统一词的使用强调即使在使用多个 IGP 和度量单位时， AS 的管理对于 AS 而言仍具有单一而一致的内部路由计划，并提供可通过它到达的网络的一致性描述。 AS 由介于 1 和 65534 之间的数字表示，并由 Internet 编号分配机构进行分配。

“**BGP**”

“**BGP**（**Border Gateway Protocol**，边界网关协议）”

一种外部网关协议，详细信息请参阅“协议概述”的 BGP 部分。

“成本”

一种 OSPF 度量单位。请参阅“度量单位”。

“延迟”

一种 HELLO 度量单位。有效值是从 0 到 30000 的数字（包括这两个数字）。值 30000 是最大的度量单位，表示不可达。请参阅“度量单位”。

“指定路由器”

OSPF: 每个多路访问网络至少具有两个连接的路由器作为一个指定路由器。指定路由器为多路访问网络生成连接状态声明，并协助运行相应协议。指定路由器由 HELLO 协议来选择。

“目标”

任何网络或任何主机。

“距离”

一种 EGP 度量单位。请参阅“度量单位”。有效值是从 0 到 255 的数字（包括这两个数字）。

“**egp**”

“**EGP**（**Exterior Gateway Protocol**，外部网关协议）”

“外部路由协议”

一种路由协议，用于在自治系统中交换路由信息。“协议概述”中提供了 **exterior gateway** 协议的详细解释。

“**EGP**”

“外部网关协议”

一种外部网关协议，详细信息请参阅“协议概述”的 EGP 部分。

“网关”

数据包发送到最终目标之前所经过的中间目标。可通过连接的网络直接到达的其他路由器的 **host** 地址。与任何主机地址一样，它可以通过符号方式指定。

“网关列表”

一个或多个由空格分隔的 **gateway**（网关）列表。

“**HELLO**”

一种内部网关协议，详细信息请参阅“协议概述”的 HELLO 部分。

“主机”

任何主机的 IP 地址。通常指定为由点号 (**.**) 分隔的四个值，这些值的范围介于 0 到 255 之间（包括这两个数字）。例如，**132.236.199.63** 或 **10.0.0.51**。它也可指定为前加 **0x** 的八位十六进制字符串。例如，**0x????????** 或 **0x0a000043**。最后，如果未指定 **options noresolv**，则可以使用符号主机名。例如，**gated.cornell.edu** 或 **nic.ddn.mil**。数字格式的优先级远远高于符号格式。

“接口”

连接的接口的 **host** 地址。它是广播、*nbma* 或环回接口的地址以及点对点接口的远程地址。与任何主机地址一样，它可以通过符号方式指定。

“接口”

路由器及其一个连接网络之间的连接。物理接口可以通过单一 IP 地址、域名或接口名指定（除非网络是未编号的点对点网络）。通过配置语言中的多个引用级别，可以使用通配符、接口类型名或删除单词地址来标识接口。由于将来的 Unix 操作系统可能允许每个接口使用多个地址，因此应谨慎使用接口名。可以添加或删除动态接口，并可将其指示为运行或关闭以及包含对地址、网络掩码和度量单位参数的更改。

“**igp**”

“内部网关协议”

“内部路由协议”

一种路由协议，用于在自治系统中交换路由信息。“协议概述”中提供了 **interior gateway** 协议的详细解释。

“接口列表”

一个或多个接口名的列表，其中包括通配符名称（不包含数字的名称）以及可指定多个接口或地址的名称（或者标识所有接口的标记 "all"）。有关详细信息，请参阅“接口列表”部分。

“**IS-IS**”

一种内部网关协议。

“本地地址”

连接的接口的 **host** 地址。它是广播、*nbma* 或环回接口的地址以及点对点接口的本地地址。与任何主机地址一样，它可以通过符号方式指定。

“掩码”

使用地址修改细分网络的一种方法。掩码是由点号分隔的四个值，它指定目标的哪些位是有效的。除了在路由过滤器中使用的情况之外，GateD 仅支持连续掩码。

“掩码长度”

掩码中有效位的数目。

“度量单位”

用于帮助系统确定最佳路由的单位之一。度量单位可以基于跃点数、路由延迟或者管理员根据路由协议类型设置的任意值。路由度量单位可能会影响分配的内部优先级的值（请参阅“优先级”）。

该示例表显示每个路由协议的可能值范围以及每个协议用来到达目标的值（请参阅“协议概述”）。

路由协议度量示例

协议	度量表示	范围	不可到达
RIP	距离（跃点数）	0-15	16
HELLO	延迟（毫秒）	0-29999	30000
OSPF	路径成本	0-?????	删除
ISIS	路径成本	0-254	删除
EGP	距离（未使用）	0-65535	255
BGP	未指定	0-65534	65535

“多路访问网络”

支持连接多个（多于两个）路由器的物理网络。假定这种网络上的每对路由器均能够直接通信。

“自然掩码”

IP 地址的格式包含网络地址和主机地址。自然掩码是应用于 3 类地址的缺省值:
0xff000000 应用于 A 类 (network.host.host.host)、
0xffff0000 应用于 B 类 (network.network.host.host) 而
0xffffff00 应用于 C 类 (network.network.network.host)。

“邻居”

通过路由协议来与之隐式或显式通信的其他路由器。邻居通常位于共享网络上，但并非始终如此。该术语最多用于 OSPF 和 EGP。通常与对等端同义。

“相邻路由器”

具有公共网络接口的两个路由器。在多路访问网络上，路由器通过 OSPF HELLO 协议动态发现。

“网络”

任何数据包交换网络。网络可以按照其 IP 地址或网络名来指定。指定网络时，主机位必须是零。可以使用 *Default* 来指定缺省网络 (0.0.0.0)。

“网络”

网络的 IP 地址。通常指定为由点号 (**.**) 分隔的一至四个值，这些值的范围介于 0 到 255 之间（包括这两个数字）。例如，**132.236.199**、 **132.236** 或 **10**。它也可以指定为前加 **0x** 的十六进制字符串，包含二到八之间的偶数位数。例如， **0x??????**、**0x????** 或 **0x0a**。此外还允许使用符号值 **default**，它具有独特值 **0.0.0.0**，即缺省网络。如果未指定 **options noresolv**，则可以使用符号网络名，如 **nr-tech-prod**、 **cornellu-net** 和 **arpanet**。数字格式的优先级远远高于符号格式。

“数字”

正整数。

“**OSPF**”

“**OSPF**（**Open Shortest Path First**，开放式最短路径优先）”

一种内部网关协议，详细信息请参阅“协议概述”的 OSPF 部分。

“**ospf_area**”

“对等端”

通过路由协议来与之隐式或显式通信的其他路由器。对等端通常位于共享网络上，但并非始终如此。该术语主要由 BGP 使用。通常与邻居同义。

“端口”

UDP 或 TCP 端口号。有效值是从 1 到 65535 的数字（包括这两个数字）。

“优先级”

preference（优先级）是介于 0（零）和 255 之间的值，用于选择到达相同目标的多条路由。具有最佳优先级（在数字上最小）的路由称作活动路由。活动路由是安装在内核转发表并导出至其他协议的路由。优先级零通常为直接连接接口的路由而保留。对于 GateD 从其中接收路由的每个源，都将分配一个缺省优先级（请参阅“优先级”）。

“前缀”

包括地址中最有意义的位的连续掩码。前缀长度指定包括多少位。

“**QoS**”

“服务质量”

TOS 在 OSI 中的等效术语。

“**RIP**”

“**RIP**（**Routing Information Protocol**，路由信息协议）”

一种内部网关协议，详细信息请参阅“协议概述”的 RIP 部分。

“路由器 **ID**”

分配给运行 OSPF 协议的每个路由器的 32 位数字。该数字唯一地标识自治系统内的路由器。

“**router_id**”

用作唯一标识符并分配来表示指定路由器的 IP 地址。它通常是所连接接口的地址。

"**RIB**"
"路由信息库"
"路由数据库"
"路由表"

GateD 保留的所有路由信息的储备库，用于做出决策，并用作所传播的路由信息的源。

"单工"

接口可能会由内核或接口配置标记单工。单工接口是无法接收所广播的数据包的广播介质上的接口。

GateD 利用能够接收其自己的广播数据包的接口来监视接口是否在正常工作。

"时间"

时间值，通常是时间间隔。它可以按以下任一格式指定：

number 秒钟的非负十进制数。例如， **27**、**60** 或 **3600**。

*number***:***number* 分钟的非负十进制数，后接处于范围 0 到 59（包括这两个数字）的秒数值。例如，**0:27**、**1:00** 或 **60:00**。

*number***:***number***:***number* 小时的非负十进制数，后接后接处于范围 0 到 59（包括这两个数字）的分钟值，再后接处于范围 0 到 59（包括这两个数字）的秒钟值。例如，**0:00:27**、**0:01:00** 或 **1:00:00**。

time to live
ttl IP 数据包的保留时间 *(TTL)*。有效值是从 1（一）到 255 的数字（包括这两个数字）。

"**TOS**"
"服务类型"

服务类型用于选择 Internet 服务质量。服务类型随着优先级、延迟、吞吐量、可靠性和成本等抽象参数一起指定。这些抽象参数将映射到数据报遍历的特定网络的实际服务参数。绝大多数的 IP 通信现在均使用缺省服务类型。

警告

gated 包含 BGP 协议规定，但是 HP 目前没有提供正式支持。本发布版中不支持这样的路由汇聚和生成语句：它们通过以显式配置方式压缩具体路由来生成更为通用的路由。

作者

请参阅 *gated*(1M) 。

另请参阅

RFC 827: E. Rosen，《Exterior Gateway Protocol EGP》

RFC 891: D. Mills，《DCN local-network protocols》

RFC 904: D. Mills，《Exterior Gateway Protocol formal specification》

RFC 1058: C. Hedrick，《Routing Information Protocol》

RFC 1105: K. Lougheed, Y. Rekhter，《Border Gateway Protocol BGP》

RFC 1163: K. Lougheed, Y. Rekhter，《A Border Gateway Protocol (BGP)》

RFC 1164: J. Honig、D. Katz、M. Mathis、Y. Rekhter、J. Yu，《Application of the Border Gateway Protocol in the Internet》

RFC 1227: M. Rose，《SNMP MUX Protocol and MIB》

RFC 1245: J. Moy，《OSPF Protocol Analysis》

RFC 1246: J. Moy，《Experience with the OSPF Protocol》

RFC 1253: F. Baker、R. Coltun，《OSPF Version 2 Management Information Base》

RFC 1256: S. Deering，《ICMP Router Discovery Messages》

RFC 1265: Y. Rekhter，《BGP Protocol Analysis》

RFC 1266: Y. Rekhter，《Experience with the BGP Protocol》

RFC 1267: K. Lougheed、Y. Rekhter，《A Border Gateway Protocol 3 (BGP-3)》

RFC 1268: P. Gross、Y. Rekhter，《Application of the Border Gateway Protocol in the Internet》

RFC 1269: J. Burruss、S. Willis，《Definitions of Managed Objects for the Border Gateway Protocol (Version 3)》

RFC 1321: R. Rivest，《The MD5 Message-Digest Algorithm》

RFC 1370: Internet 体系结构委员会，《Applicability Statement for OSPF》

RFC 1388: G. Malkin，《RIP Version 2 Carrying Additional Information》

RFC 1397: D. Haskin，《Default Route Advertisement In BGP2 And BGP3 Versions Of The Border Gateway Protocol》

RFC 1403: K. Varadhan，《BGP OSPF Interaction》

RFC 1583: J. Moy，《OSPF Version 2》

名称

gettydefs - getty 所使用的速率和终端设置

说明

/etc/gettydefs 文件包含 **getty** 为一行设置线路的速率和终端设置时所使用的信息（请参阅 *getty*(1M) ）。它还说明了 **login** 提示信息的格式。如果用户键入 **Break** 字符表明当前速率不正确，这个文件还为用户提供可以尝试的其他速率。

/etc/gettydefs 中每个条目的格式如下所示：

label# *initial-flags* # *final-flags* # *login-prompt* #*next-label*

每个条目后都接一个空行。各个字段具有 **\b** 、 **\n** 、 **\c** 以及 *****nnn* 形式的引用字符，在这里， *nnn* 是字符的八进制值。各个字段为：

label **getty** 使用本字符串与其第二个参数进行匹配。它通常是终端所运行的速率，如 **1200** ，但不一定是这个速率，请参阅下面的内容。

initial-flags 如果没有为 **getty** 指定终端类型，这些标志就是需要设置的终端的初始 **ioctl()** 设置。请参阅（ *ioctl*(2) ）。 **getty** 所理解的标志与 **/usr/include/sys/termio.h** 中所列的标志是相同的（请参阅 *termio*(7) ）。通常情况下， *initial-flags* 只需要速率标志， **getty** 自动将终端设置为原始输入模式，并处理其他大多数标志。在 **getty** 执行 **login** 之前， *initial-flag* 设置是有效的。

final-flags 这些标志的取值与 *initial-flags* 相同，而且是在 **getty** 执行 **login** 之前设置的。这里也需要速率标志。复合标志 **SANE** 处理其他大多数需要设置的标志，使处理器与终端以合理的方式进行通信。通常还指定另外两个 *final-flags* 标志，即 **TAB3** 和 **HUPCL** 标志。前一标志用于将制表符作为空格发送给终端，后一标志用于在最后关闭设备时挂起线路。

login-prompt 将整个字段作为 *login-prompt* 打印出来。在上面的字段中，空白字符（空格、制表符和换行符等）被忽略，但是在这里，它们包含在 *login-prompt* 字段中。

next-label 这个条目并不指定所需的速率，用户输入 **Break** 字符后， **getty** 查找 *label* 字段为 *next-label* 的条目，并使用这些条目中指定的设置来配置终端。通常情况下，一系列的速率以这种方式链接在一个封闭集合中。例如， **2400** 与 **1200** 链接，而 1200 又与 **300** 链接，而 300 最终又与 **2400** 链接。

如果调用 **getty** 时没有使用第二个参数，将使用 **/etc/gettydefs** 中的首个条目，从而使 **/etc/gettydefs** 中的首个条目成为缺省条目。如果 **getty** 不能查找到指定的 *label* 条目，也将使用此条目。如果缺少 **/etc/gettydefs** ，则命令内置一个条目，使终端以 **300** 波特率启动。

强烈建议在创建或修改 **/etc/gettydefs** 之后，运行带有检查选项的 **getty** ，以确保没有错误。

举例

以下行是一个 300/1200 波特率切换示例，它用于拨号端口：

1200# B1200 HUPCL # B1200 SANE IXANY IXANY TAB3 #login: #300
300# B300 HUPCL # B300 SANE IXANY IXANY TAB3 #login: #1200

以下行显示硬连接的典型 9600 波特率条目:

9600# B9600 # B9600 SANE IXANY IXANY ECHOE TAB3 #login: #9600

文件

/etc/gettydefs

另请参阅

getty(1M)、 login(1)、 ioctl(2)、 termio(7)。

名称

group、logingroup - 组文件，grp.h

说明

group 包含每个组的以下信息：

- 组名
- 加密口令
- 数字组 ID
- 以逗号分隔的组所有允许用户列表，

这个文件是 ASCII 文件。各个字段之间用冒号分隔，各个组之间用换行符分隔。不能用空格分隔行上的各个字段以及字段的各组成部分。如果口令字段为空，则没有与组相关联的口令。

在系统中，这种形式的文件有两个： **/etc/group** 和 **/etc/logingroup** 。文件 **/etc/group** 为每个组提供名称，并支持通过 **newgrp** 实用程序（请参阅 *newgrp*(1) ）进行的组更改； **/etc/logingroup** 通过 **login** 和 **initgroups()** （请参阅 *login*(1) 和 *initgroups*(3C) ）向每个用户提供缺省组访问列表。

login 为每个用户创建的真实有效的组 ID 定义在 **/etc/passwd** 之中（请参阅 *passwd*(4) ）。如果 **/etc/logingroup** 为空，缺省组访问列表也为空。如果 **/etc/logingroup** 和 **/etc/group** 指向同一个文件链接，缺省访问列表将包含与用户相关联的整个组集。将不使用 **/etc/logingroup** 中的组名和口令字段；这些组名和口令字段仅用于为这两个文件提供统一的格式，从而允许链接它们。

/etc/logingroup 或 **/etc/passwd** 中使用的所有 ID都应定义在 **/etc/group** 之中。对于一个用户，与其相关联的组的数量不能超过 **/etc/logingroup** 中的 **NGROUPS** （请参阅 *setgroups*(2) ）。

这些文件位于目录 **/etc** 。由于使用了加密口令，这些文件可以具有而且的确具有一般读取权限，例如，用于将数字组 ID映射为名称。

组结构定义在 **<grp.h>** 之中，并包含以下成员：

```
char    *gr_name;     /* the name of the group */
char    *gr_passwd;   /* the encrypted group password */
gid_t   gr_gid;       /* the numerical group ID */
char    **gr_mem;     /* null-terminated array of pointers to member names */
```

网络功能

NIS

/etc/group 文件可以包含以加号 **(+)** 开头的行，它表示合并网络信息服务 (NIS) 中的条目。有两种形式的 **+** 条目： **+** 表示在该点插入 NIS 组文件的整个内容， **+***name* 表示在该点插入 NIS 中的 *name* 条目（如果有）。如果 **+** 条目有一个非空口令或组成员字段，字段的内容将覆盖 NIS 中包含的内容。数字组 ID 字段不能被覆盖。

组文件可以包含以减号 **(-)** 开头的行，这些条目用于禁用组条目。 **-** 条目只有一种形式；任何包含 **-***name* 的条目都表示禁用 *name* 的后继条目（如果有）。这些条目都被禁用，无论后继条目来自 NIS 或本地组文件。

警告

组文件不能包含任何空白行。空白行将导致使用这些文件的系统管理软件发生不可预知的行为。

组 ID (*gid*) 9 是为 Pascal 语言操作系统和 BASIC 语言操作系统保留的。这些操作系统是 Series 300/400 计算机所使用的操作系统，它们可以与 HP-UX 在同一磁盘上共存。将此 gid 用作其他目的将禁止文件传输和共享。

/etc/group 中每行的长度不能超过 **<limits.h>** 中定义的 **LINE_MAX** 。由于有这个限制，因此不应在主要组中列出用户，而只应在附加组中列出。

如果 **/etc/group** 链接到 **/etc/logingroup** ，用户的组成员关系由 NIS 管理，而且没有能够作出响应的 NIS 服务器，则用户不能登录，直到有服务器作出响应。

没有任何工具可以确保 **/etc/passwd** 、 **/etc/group** 和 **/etc/logingroup** 兼容。但是，使用 **pwck** 和 **grpck** 可以简化任务（请参阅 *pwck*(1M) ）。

使用任何工具都无法设置 **/etc/group** 中的组口令。

相关内容

NIS

举例

以下是一个简单的 **/etc/group** 文件：

```
other:*:1:root,daemon,uucp,who,date,sync
-oldproj
bin:*:2:root,bin,daemon,lp
+myproject:::bill,steve
+:
```

组 **other** 的 GID 为 1，它还有 **root** 、 **daemon** 、 **uucp** 、 **who** 、 **date** 和 **sync** 等成员。组 **oldproj** 被忽略，因为它位于条目 **-oldproj** 之后。此外，组 **myproject** 有成员 **bill** 和 **steve** ，还有口令以及 **myproject** 组的 NIS 条目的组 ID 。 NIS 中列出的所有组被引用，并位于 **myproject** 之后。

警告

加号 (**+**) 和减号 (**-**) 功能是 NIS 的一部分。因此，如果没有安装 NIS ，该功能将不起作用。

文件

/etc/group
/etc/logingroup

另请参阅

groups(1)、 newgrp(1)、 passwd(1)、 setgroups(2)、 crypt(3C)、 getgrent(3C)、 initgroups(3C)、 passwd(4)。

符合的标准

group：SVID2、SVID3、XPG2

名称

hosts - 主机名数据库

说明

/etc/hosts 文件将 Internet (IP) 地址与正式主机名和别名相关联。这就允许用户使用符号名称而不是 Internet 地址来指向一个主机。

注释：此文件 必须包含 **ifconfig** 在引导时所需的所有本地接口地址（请参阅 *ifconfig*(1M) ）。当使用名称服务器（请参阅 *named*(1M) ）或网络信息服务（请参阅 *ypserv*(1M) ）时，此文件经常用作服务器没有运行时的一个备份。在这种情况下，这是 **/etc/hosts** 包含本地网络中一些计算机地址的通常做法。

/etc/hosts 应为每个主机包含一行下述信息：

internet_address official_host_name aliases

internet_address 可以是一个以传统 Internet 点分表示法指定的 IPv4 或 IPv6 地址。有关 Internet 地址处理例行程序的更多信息，请参阅 *inet*(3N) 或 *inet6*(3N) 。

aliases 是用于识别一个已知主机的其他名称。在大多数命令中，它们可替代 *official_host_name* 。例如：

192.45.36.5 hpdxsg testhost

在此示例中，用户可通过使用下述命令来进行远程登录 **hpdxsg** ：

rlogin testhost

而不是

rlogin hpdxsg

如果系统处于域命名环境下，则正式主机名包含完整的域扩展主机名。例如：

192.45.36.5 hpdxsg.xsg.hp.com hpdxsg testhost

行不能以空格开头（空格字符或制表符）。各项目由任意数量的空格字符和制表符（空白）或其组合来分隔。 **#** 字符表示注释的开头。搜索该文件的例行程序不解释从 **#** 到行尾之间的字符。行的结尾允许有结尾空白字符。

对于 Internet，尽管可能需要对此文件进行一些本地更改，从而使其成为关于非正式别名和（或）未知主机的最新文件，但它通常是从在网络信息控制中心 (NIC) 中保留的正式主机数据库中创建的。

主机名可包含任意一个可输出字符，但不能包括空白字符、换行符或注释字符。

举例

请参阅 **/etc/hosts** 。

作者

hosts 由加州大学伯克利分校开发。

另请参阅

gethostent(3N)、 inet(3N)、 nsswitch.conf(4)。

名称

hosts.equiv、.rhosts - 授权远程主机和本地主机用户访问的安全文件

说明

在用户主目录的 **/etc/hosts.equiv** 文件和名为 **.rhosts** 的文件指定了“等价于”本地主机或者用户的远程主机和用户。来自等价的远程主机的用户允许使用 **rcp** 或者 **remsh** 访问一个本地帐户或者 **rlogin** 到本地帐户而无须提供口令（请参阅 *rcp*(1) 、 *remsh*(1) 和 *rlogin*(1) ）。由 **hosts.equiv** 所提供的安全性通过 **ruserok()** 库例行程序实现，（请参阅 *rcmd*(3N) ）。

在本说明中， *hostequiv* 表示系统 **/etc/hosts.equiv** 文件或者用户 **.rhosts** 文件。请注意， **.rhosts** 必须由 root 所有或者由其所在主目录相应的用户所有，并且它必须不是符号链接。 **/etc/hosts.equiv** 文件定义了系统级内的等价性，而用户的 **.rhosts** 文件定义了本地用户与任意远程用户的等价性，其中对该远程用户的访问本地用户选择允许或者拒绝。

hostequiv 文件中的一个条目是有如下格式的一行（没有延续）：

> [*hostname* [*username*]] [**#***comment*]

因此，它可以是:

- 空白行。
- 注释行，以 **#** 开头。
- 主机名，可以选择后面跟有注释。
- 主机名和用户名，可以选择后面跟有注释。

主机名称或者用户名称是一个可打印字符串，不包含空白字符、换行符和 **#** 。

名称以空白字符分开。

要授予一个用户访问权限，远程主机名和用户名都必须“匹配” *hostequiv* 中的一个条目。当发出一个访问请求时，会首先搜索 **/etc/hosts.equiv** 文件。如果发现了匹配，就允许访问。在用户的主目录中有此文件的情况下，如果没有匹配，则会查找 **.rhosts** 文件。如果本地用户是超级用户，则 **/etc/hosts.equiv** 被忽略。

主机名或者用户名称必须以下列方式之一匹配 *hostequiv* 中的相应字段条目:

Literal match

hostequiv 中的主机名可以严格匹配远程主机的正式主机名（非别名）。

hostequiv 中的用户名可以严格匹配远程用户名。要使一个用户名能够在 **/etc/hosts.equiv** 文件中有严格的匹配，远程用户名必须严格地匹配本地用户名。

Domain-extended match

要与 *hostequiv* 中的条目相比较的远程主机名称通常是由 **gethostbyaddr()** 返回的标准主机名称（请参阅 *gethostent*(3N) ）。在一个域名命名环境中，这是一个域名限定的名称。如果 *hostequiv* 中的一个主机名不能与远程主机名严格的匹配，则在 *hostequiv* 中的主机名后加上本地域名可能会匹配远程主机名。

-*name*

如果 *hostequiv* 中的主机名是这样的形式，并且如果 *name* 严格匹配远程主机名或者如果 *name* 后面加上本地域名匹配远程主机名，则不论用户名是什么都拒绝访问。

如果 *hostequiv* 中的用户名是这样的形式，并且 *name* 严格匹配远程用户名，则访问被拒绝。

即使访问被 **/etc/hosts.equiv** 以此种方式拒绝，还可以通过 **.rhosts** 进行访问。

+

任何远程主机名匹配 *hostequiv* 中的主机名 **+** 。

任何的远程用户匹配用户名 **+** 。

+@*netgroup_name*

netgroup_name 是定义在 *netgroup*(4) 中的网络组名。如果 *hostequiv* 中的主机名是这种形式，则要使主机名匹配，远程主机名必须（仅）根据在 *netgroup*(4) 中定义的规则匹配指定的网络组。

类似地，如果 *hostequiv* 中的用户名是这种形式，要使用户名匹配必须（仅）远程用户名匹配指定的网络组。

-@*netgroup_name*

netgroup_name 是定义在 *netgroup*(4) 的网络组名称。如果 *hostequiv* 中的主机名是这种形式，并且根据定义在 *netgroup*(4) 中的规则，如果远程主机名（仅）匹配指定的网络组，则访问被拒绝。

类似地，如果定义在 *hostequiv* 中的用户名是这种形式，并且如果远程用户名（仅）匹配指定的网络组，则访问被拒绝。

即使访问被 **/etc/hosts.equiv** 以这种方式拒绝，仍然可以通过 **.rhosts** 允许访问。

举例

1. **/etc/hosts.equiv** 位于 **hostA** ，它包含此行：

   hostB

   并且 **/etc/hosts.equiv** 位于 **hostB** ，为空。 **hostB** 上面的用户 **chm** 可以使用 **remsh** 到 **hostA** ，或者使用 **rlogin hostA** 上面的 **chm** 帐户而无需提示输入口令。然而， **chm** 能够使用 **rlogin** 来提示口令或者使用 **remsh** 来拒绝 **hostA** 到 **hostB** 的访问。

   如果 **hostB** 上的用户 **chm** 主目录中的 **.rhosts** 包含了：

   hostA

   或者

   hostA chm

那么用户 **chm** 可以从 **hostA** 访问 **hostB** 。

2. **hostA** 位于域 **arg.bob.com** 中。 **hostB** 和 **hostC** 位于域 **oink.bob.com** 中。 **hostB** 上的用户 **chm** 主目录中的 **.rhosts** 包含:

   hostC
   hostA

   用户 **chm** 可以从 **hostC** 访问 **hostB** ，因为当 **hostB** 后面加上本地域 **oink.bob.com** 时，**hostC.oink.bob.com** 匹配 **hostC** 。但是来自 **hostA** 的用户 **chm** 不能访问 **hostB** ，因为 **hostA.arg.bob.com** 不匹配 **hostA.oink.bob.com** 。要使得用户 **chm** 能够从 **hostA** 访问 **hostB** ， **hostB** 上 **chm** 的 **.rhosts** 文件必须包含:

   hostA.arg.bob.com

   因为 **hostA** 在不同的域中。

3. **hostA** 上用户 **chm** 主目录中的 **.rhosts** 包含:

   hostB root

   hostB 上的 **/etc/hosts.equiv** 包含此行:

   hostA

   然而，在 **hostB** 上的用户 **chm** 主目录中不存在文件 **.rhosts** 。 **hostB** 上的用户 **root** 可以 **rlogin hostA** 上的 **chm** 帐户而无需提示输入口令，但是 **hostA** 上的 **root** 不能 **rlogin hostB** 上的 **chm** 上的帐户。

4. **hostA** 上的用户 **chm** 主目录中 **.rhosts** 包含了:

   +
   -hostB
   + root

   来自任何主机上的用户 **chm** 都允许访问 **hostA** 上的帐户 **chm** 。 除了 **hostB** ，任何主机的用户 **root** 都可以访问 **hostA** 上的帐户 **chm** 。

5. **hostA** 上的 **/etc/hosts.equiv** 包含下面几行:

   + -chm
   hostB

   除了 **chm** 任何 **hostB** 上的用户都允许使用同样的用户名访问 **hostA** 上的帐户。然而，如果 **hostA** 上的用户 **chm** 主目录中的 **.rhosts** 包含:

   hostB

   然后， **hostB** 上的用户 **chm** 可以访问 **hostA** 上的帐户 **chm** 。

6. **hostA** 上的 **/etc/hosts.equiv** 包含下面一行：

   +@example_group

   网络组 **example_group** 包含了：

   example_group (, ,EXAMPLE_DOMAIN)

   如果 **hostA** 没有运行网络信息服务 (NIS)，任何主机上的用户 **chm** 都可以访问 **hostA** 上的帐户 **chm** 。

   如果 **hostA** 运行了网络信息服务 (NIS)，并且 **hostA** 在域 **EXAMPLE_DOMAIN** 中，任何主机上的用户 **chm** ，不论是否在 **EXAMPLE_DOMAIN** 中都能访问 **hostA** 上的帐户 **chm** 。

   然而，如果 **hostA** 上的用户 **chm** 主目录中的 **.rhosts** 包含下面一行：

   -@example_group

   而且 **hostA** 或者没有运行网络信息服务 (NIS)，或者在域 **EXAMPLE_DOMAIN** 中时，任何主机上的用户 **chm** 都能访问 **hostA** 上的帐户 **chm** 。 如果 **hostA** 正在运行网络信息服务 (NIS) 但并不在域 **EXAMPLE_DOMAIN** 中，此行无效。

7. **hostA** 上的 **/etc/hosts.equiv** 包含下面一行：

   -@example_group

   网络组 **example_group** 包含了：

   example_group (hostB, ,)

   hostB 上的所有用户都被拒绝访问 **hostA** 。

   然而，如果 **hostA** 上用户主目录中的 **.rhosts** 包含了下面任何一行：

   +@example_group chm
   hostB chm
   + chm

   那么 **hostB** 上的用户 **chm** 可以访问 **hostA** 的帐户。

警告

出于安全考虑，文件 **/etc/hosts.equiv** 和 **.rhosts** 应该存在并只能被其所有者读写，即使它们为空。

创建 **/etc/hosts.equiv** 时必须小心

remshd 和 **rlogind** 的 **-l** 选项根据 **.rhosts** 文件防止除了超级用户以外的任何验证。

作者

hosts.equiv 由加州大学伯克利分校开发。

+ 、 **-***name*, **+@***netgroup_name*, 和 **-@***netgroup_name*, 扩展由 Sun Microsystems, Inc 开发。

文件

$HOME/.rhosts

/etc/hosts.equiv

另请参阅

rcp(1)、 rdist(1)、 remsh(1)、 rlogin(1)、 remshd(1M)、 rlogind(1M)、 gethostent(3N)、 rcmd(3N)、 netgroup(4)。

名称

inetd.conf - inetd 的配置文件

说明

inetd 守护程序被调用时将从 **/etc/inetd.conf** 配置文件读取信息，它也可能在稍后某个时间为响应 **SIGHUP** 信号而读取此配置文件（请参阅 *inetd*(1M)）。

文件中的每一行都被视为注释或指定服务的配置信息。以 **#** 开头的行表示注释。根据第一个字段中指定的服务名称，非注释行包含七个或九个必需的字段。字段之间用制表符和（或）空格分隔。可以用 **** 将一行分为多个连续行。文件中的每个配置行依次包含以下字段：

- *service name*
- *socket type*
- *protocol*
- **wait|swait|nowait**
- *user*
- *server program*
- *program number* （仅适用于 NFS RPC 服务）
- *version number* （仅适用于 NFS RPC 服务）
- *server program arguments*

字段的构建方式如下所示：

service name — **rpc** 如果服务器基于 RPC (NFS)；否则将是文件 **/etc/services** 中有效服务的名称。例如，**remsh** 服务（请参阅 *remsh*(1)）的 **shell**、**rlogin** 服务（请参阅 *rlogin*(1)）的 **login** 和 **telnet** 服务（请参阅 *telnet*(1)）的 **telnet**。

socket type — **stream**、**dgram** 或 **xti** 取决于服务器套接字是流套接字、数据报套接字，还是应用于使用 XTI API 构建的程序。

protocol — 必须是 **/etc/protocols** 中给定的有效协议；例如 **tcp** 或 **udp**。如果 XTI 是在套接字类型字段中指定的，那么在这里必须指定设备的完整路径名，如 **/dev/tcp**；否则，这里指定的协议将被追加到 **/dev/**。例如，如果为 XTI 应用程序指定了 **tcp**，将使用路径 **/dev/tcp**。

对于 IPv6 应用程序，协议指定为 **tcp6** 或 **udp6**。

wait|swait|nowait — 指定 **inetd** 用作单线程服务器或多线程服务器。

wait — 指示 **inetd** 启动服务器，以处理传入的请求，并停止监听来自同一服务器的新请求，直到已启动的服务器退出。

swait 与 **wait** 相同，但指示 **inetd** 以要求服务器接受传入的请求。

nowait 指示 **inetd** 启动一个服务器，以处理所有传入请求。

对于大多数基于 UDP 的服务，这个字段都使用 **wait** ，而基于 TCP 的服务则使用 **nowait** 。

user 服务器运行时所使用的用户 ID。

server program 当 **inetd** 发现服务器套接字上的请求时执行的程序的绝对路径名。

server program arguments
服务器程序的参数。与标准用法相同，以 **argv[0]** （程序名称）开头。

如果 *service name* 是 **rpc** （NFS RPC 服务），则需要两个额外的字段。它们必须位于 *server program* 字段与 *server program arguments* 字段之间：

program number 定义特定服务分组，它是唯一的。

version number RPC 服务所支持的版本。如果程序处理多个版本，这个编号可以是一个值，也可以是一个范围；例如， **1** 或 **1-3** 。范围用连字符 **(-)** 分隔。版本号允许 RPC 协议被扩展和修改，而且使旧版协议和新版协议共享同一个服务器进程成为可能。

内置 inetd 服务

通过使用内置例行程序， **inetd** 守护程序提供多个“普通”服务（有关这些服务的列表请参阅 *inetd*(1M) ）。若要配置内部服务，请将 **internal** 指定为 *server program* 名称，并忽略 *server program arguments* 字段。

举例

配置 **shell** 服务，以使用 TCP 协议，并以 **root** 用户运行服务器 **remshd** 。

shell stream tcp nowait root /usr/lbin/remshd remshd

以上是 IPv4 模式下运行的 remsh 实用程序示例。若要在 IPv6 模式下运行以上实用程序，协议 **tcp** 必须更改为 **tcp6** 。为了能够在 **IPv6** 模式下运行，需要重写上面的配置，如下所示：

shell stream tcp6 nowait root /usr/lbin/remshd remshd

配置 FTP 服务器，使非活动会话在 75 秒后超时。

ftp stream tcp nowait root /usr/lbin/ftpd ftpd -t75

利用下面的配置，可以使上面的 ftp 服务在 IPv6 模式下运行：

ftp stream tcp6 nowait root /usr/lbin/ftpd ftpd -t75

配置基于 RPC 的服务。请注意， *service name* 字段包含 **rpc** 并使用另外两个字段：程序编号 (100008) 和版本号 (1)。

rpc dgram udp wait root /usr/lib/netsvc/rwall/rpc.rwalld 100008 1 rpc.rwalld

配置 **inetd** ，以使用内置 **daytime** 的 TCP 服务。

daytime stream tcp nowait root internal

警告

安装可选的 IPv6 软件后，HP-UX 11i V1.0 支持 IPv6。目前，运行 HP-UX 11i V1.6 的系统不支持 IPv6。

作者

inetd.conf 由加州大学伯克利分校开发。

NFS 由 Sun Microsystems, Inc. 开发。

另请参阅

inetd(1M)、 exec(2)、 fork(2)、 inetd.sec(4)、 protocols(4)、 services(4)。

名称

inetd.sec - inetd 的可选安全文件

说明

当 **inetd** 接受来自远程系统的连接时，它将请求服务的主机的地址与特定服务的允许访问或拒绝访问主机列表进行比较检查（请参阅 *inetd*(1M) ）。文件 **inetd.sec** 允许系统管理员控制哪些主机（或一般网络）可以远程使用系统。这个文件形成了除服务所完成的标准检查之外的另一个额外的安全层。它处理服务器的安全问题，也就是说，服务器不能被 Internet 守护程序启动，除非请求服务的主机是受到 **inetd.sec** 许可的有效主机。

如果没有文件 **/var/adm/inetd.sec** ，则只有服务器所实现的安全。 **inetd.sec** 和目录 **/var/adm** 应只能被其所有者进行写操作。 **inetd.sec** 的更改将应用于以后的连接。

在 **inetd.sec** 中以 **#** 开头的行是注释。数据行的末尾不允许有注释。

文件中的行包含服务名称、权限字段和允许使用本地主机上的服务的主机和网络的 Internet 地址或标准名称。每行中的字段如下所示：

<*service name*> <**allow|deny**> <*host/net addresses, host/net names*>

service name 是文件 **/etc/services** 中有效服务的名称（非别名）。基于 RPC 的服务 (NFS) 是文件 **/etc/rpc** 中有效服务的名称（非别名）。 **/etc/rpc** 中的服务名称对应于一个唯一的 RPC 程序编号。

allow|deny 确定允许或拒绝下一字段中的远程主机列表访问特定的服务。不支持每个服务的多个 **allow|deny** 行。如果特定的服务有多行 **allow|deny** ，将忽略除最后一行以外的所有行。

地址和名称用空格字符分隔。允许任意组合地址和名称。若要延续行，应使用 **** 来结束行。

gethostbyaddr() 或 **getnetbyaddr()** 分别返回主机名和网络名，它们都是标准名称。允许使用通配符 (*****) 和范围符 (**-**) 。 ***** 和 **-** 可以位于地址字段的任意位置。地址字段是用点 (**.**) 分隔的字符串。

举例

使用通配符，以允许整个网络与本地主机通信，而无须列出网络中的所有主机。例如，若要允许网络地址以 **10** 开头的所有主机以及地址为 **192.54.24.5** 的主机使用 *rlogin* ：

```
login     allow   10.*  192.54.24.5
```

在运行 NFS 的系统上，拒绝主机 **192.54.24.5** 访问基于 RPC 的服务器 *sprayd* ：

```
sprayd    deny    192.54.24.5
```

range 是一个包含了 **-** 字符的字段。拒绝网络 10 (arpa) 中的子网 3 到子网 5 中的主机访问 **remsh** ：

```
shell     deny    10.3-5.*
```

下面的条目拒绝 **rlogin** 访问主机 **cory.berkeley.edu** 、名称为 **testlan** 的网络上的任何主机以及 Internet 地址为 **192.54.24.5** 的主机：

```
login     deny    192.54.24.5 cory.berkeley.edu testlan
```

如果远程服务器没有在安全文件中列出，或者如果已经列出但其后不接 **allow** 或 **deny** ，那么所有的远程主机都

可以尝试使用它。这时，安全由服务本身提供。以下行（如果位于 **inetd.sec** 之中）允许或拒绝访问已指明的服务：

允许主机使用 **ftp** ：

ftp

拒绝对 **shell** 服务（如 **remsh** ）的所有访问：

shell deny

允许访问由任何主机提供的 **shell** 服务：

shell allow

或者

shell

IPv6 功能

对于 IPv6 服务，可以在 **inetd.sec** 的主机地址字段中指定 IPv6 地址。主机地址字段可以包含 IPv6 地址、IPv4 地址，也可以同时包含这两类地址。这个规范还包含 IPc4 映射为 IPv6 的地址。

getaddrinfo() 返回的正式主机名是 IPv6 服务的主机名。

IPv6 地址不支持通配符 (*) 和范围符 (-) 。但提供了通配符 (*) 的等同形式，即 **subnet_prefix** 后接反斜线 (/) 和 **prefix_length** 。有关详细信息，请参阅 IPv6 举例一节。

IPv6 举例

为了使地址为 **fe80::210:83ff:feb9:903f** 的 IPv6 主机和地址为 **192.54.24.5** 的 IPv4 主机使用 **telnet** 服务， **inetd.sec** 文件中应有以下条目：

telnet allow fe80::210:83ff:feb9:903f 192.54.24.5

以下条目拒绝 **ftp** 访问所有前缀为 **fe80** 的主机：

ftp deny fe80::/16

警告

如果安装了可选的 IPv6 软件，HP-UX 11i v1.0 支持 IPv6。目前，运行 HP-UX 11i v1.6 的系统不支持 IPv6。

作者

inetd.sec 由 HP 开发。

NFS 由 Sun Microsystems, Inc. 开发。

文件

/var/adm/inetd.sec

另请参阅

inetd(1M)、 gethostent(3N)、 getaddrinfo(3N)、 getnetent(3N)、 hosts(4)、 inetd.conf(4)、 networks(4)、 protocols(4)、 rpc(4)、 services(4)。

名称

inetsvcs.conf - 安全 Internet 服务的配置文件

说明

Internet 服务、ftp、rcp、remsh、rlogin 和 telnet 使用 /etc/inetsvcs.conf 配置文件来决定它们的行为（即是否允许使用 Kerberos V5 的网络验证）。文件的内容决定安全 Internet 服务是否启用。此配置文件通过程序 **inetsvcs_sec** 更新。文件中的缺省条目如下所示：

kerberos false

使用此条目，所有指定的服务显示它们的传统行为（即通过提示用户输入口令来提供验证）。

要启用安全 Internet 服务，inetsvcs_sec 程序被用于使用下列条目更新配置文件：

kerberos true

警告

文件不应手动更新。如果无效的条目存在此文件中，服务不会运行。程序 **inetsvcs_sec** 必须用于更新配置文件。

另请参阅

sis(5)、inetsvcs_sec(1M)。

名称

info - 无盘客户端配置信息文件

说明

info 文件是 POSIX Shell 资源文件，它包含引导时所使用的参数定义。一般情况下，它是一个空文件，所有参数都使用缺省值。以下是可以在 **info** 文件中定义的参数的列表：

ROOT_SERVER_IP 指定客户端的私有根服务器的 IP 地址。如果没有指定，客户端的私有根服务器缺省为引导服务器。

PRIVATE_ROOT 指定私有根服务器上客户端私有根目录的路径名。如果没有指定，客户端的私有根目录路径缺省为 **/export/private_roots/***client_name* 。

MOUNT_ROOT_OPTS 指定 NFS 挂接选项，从私有根服务器挂接客户端的私有根目录。如果没有指定，挂接选项缺省为 **boot,hard,nointr,nodevs** 。

MOUNT_STAND_OPTS 指定 NFS 挂接选项，从引导服务器挂接客户端的 **/stand** 目录。如果没有指定，挂接选项缺省为 **boot,hard,nointr,nodevs** 。

NO_SWAP_TO_NFS 指定是否将 NFS 配置为主交换分区（注释：为了与 NFS 进行交换，必须使用设置为 1 的可调参数 **remote_nfs_swap** 配置无盘内核）。如果无盘计算机有本地交换磁盘，而且不必与 NFS 交换，则 **NO_SWAP_TO_NFS** 参数应设置为值 1，应配置无盘内核而不必将 **remote_nfs_swap** 设置为 1。如果 **info** 文件中没有指定此参数，而且内核可调参数 **remote_nfs_swap** 设置为 1，那么 NFS 将被配置为主交换分区。

REMOVE_EXTRA_SWAPFILES
如果未设置，则参数的缺省值为值 1；无盘客户引导时，将删除所有超过已配置的最小交换的（交换 **min** 是在客户端的 **/etc/fstab** 中指定的）所有交换文件。这确保在引导时删除多余的交换文件，从而释放磁盘空间。如果 **info** 文件中的 **REMOVE_EXTRA_SWAPFILES** 设置为 0，将禁止删除多余的交换文件。这可能使得引导速度更快，因为无须创建额外的交换文件。

info 文件与客户端内核一样，都位于引导服务器上的 (**/export/tftpboot/***client***/stand**) 目录，在引导时，可以使用 **tftp** 命令检索该文件。缺省情况下，当创建一个无盘客户端时，将在客户端的内核目录下存放一个空的 **info** 文件。这就确保所有参数都恢复为缺省值（请参阅前面的内容）。如果没有此文件，将产生错误。

举例

以下是一个 **info** 文件示例：

```
# Sample info(4) file:
# set NO_SWAP_TO_NFS
NO_SWAP_TO_NFS=1
```

文件

/export/tftpboot/*client***/stand/info**

名称

inittab - 引导初始化进程的脚本

说明

/etc/inittab 文件提供了脚本来引导 **init** 守护程序，其角色相当于通用进程调度程序（请参阅 *init*(1M)）。构成引导 **init** 的进程调度活动的主要进程是线进程 **/usr/sbin/getty**，它单个启动各终端线。其他由引导 **init** 所调度的典型进程通常是守护程序和 Shell。

inittab 文件由位置相关的条目组成，并且这些条目具有如下格式：

*id***:***rstate***:***action***:***process*

每一个条目由一个换行符分隔；然而，在换行符之前有一个反斜线 (\)，这表示该条目的延续。每个条目至多允许有 1024 个字符。通过在“单词”前加一个 **#**，即可在 *process* 字段中插入注释（请参阅 *sh*(1)）。产生 **getty** 的行注释可通过 **who** 命令来显示（请参阅 *who*(1)）。它们可能会包含一些像行位置这样的行信息。**inittab** 文件中条目的数目没有限制（只有最大条目大小）。

条目字段是：

id 一个一到四个字符的值，用于唯一标识一个条目。条目重复会导致发出一个出错消息，而其他情况将忽略。强烈推荐使用一个四字符的值来标识一个条目（请参阅下面的警告）。

rstate 定义要处理此条目的位置 *run level*。运行级别对应系统中进程的一个配置，其中每个由引导 **init** 产生的进程都被赋予一个或者多个其得以存在的运行级别。运行级别由从 **0** 到 **6** 范围内的一个数表示。例如，如果系统处于运行级别 **1**，则只有那些在 *rstate* 字段中有 **1** 的条目才会被处理。

当请求引导 **init** 改变运行级别时，所有在 *rstate* 字段中没有目标运行级别条目的进程都会收到一个警告信号 (**SIGTERM**)，并有 20 秒时间可以从容退出，否则进程将被 kill 信号强制结束 (**SIGKILL**)。您可通过在任意组合中输入多个运行级别值来为一个进程指定多个运行级别。如果不指定运行级别，则假设进程能在所有运行级别上有效，从 **0** 到 **6**。

其他三个值，**a**、**b** 和 **c** 也可以出现在 *rstate* 字段中，尽管它们不是真正的运行级别。在 *rstate* 字段中有这些字符的条目只有当一个用户 **init** 进程请求它们运行时才被处理（不论当前的运行级别是什么）。它们和运行级别的区别在于引导 **init** 永远不会输入“运行级别” **a**、**b** 或者 **c**。并且，请求执行这些进程中的任意一个不会改变当前的数字运行级别。

而且，一个由 **a**、**b** 或者 **c** 启动的进程在引导 **init** 改变运行级别时不会结束。只有一个进程的行在 **inittab** 中的 *action* 字段被标记为 **off** 时、完全从 **inittab** 删除改行或者引导 **init** 进入 *single-user* 状态时，进程才会结束。

action 此字段中的一个关键字告诉引导 **init** 如何处理在 *process* 字段中指定的进程。可以指定下列操作：

boot 只有在引导 **init** 的引导过程中读取 **inittab** 文件时处理该条目。引导 **init** 启动进程，不等待其结束，并在进程结束时不重启进程。要使此说明有

意义， *rstate* 应当是缺省的，或者是在引导时间里必须匹配引导 **init** 的运行级别。此操作对于硬件引导系统后的初始化功能很有用。

bootwait 只有在引导 **init** 的引导过程中读取 **inittab** 文件时处理该条目。引导 **init** 启动进程，等待其结束，并且在进程结束时不重启进程。

initdefault 仅在引导 **init** 最初调用时，才扫描具有此 *action* 的一个条目。如果此条目存在，引导 **init** 将使用其确定最初进入哪个运行级别。通过采用在 **rstate** 字段中指定的最高运行级别，并使用其作为自己的初始状态来实现此操作。如果 *rstate* 字段为空，则引导 **init** 进入运行级别 **6** 。

initdefault 条目不能指定使引导 **init** 以单用户状态启动。此外，如果引导 **init** 没有在 **inittab** 中找到 **initdefault** 条目，则在启动时向用户请求一个初始运行级别。

off 如果与此条目相关的进程现在正在运行，则发送警告信号 (**SIGTERM**) 并等待 20 秒，然后就可以通过 kill 信号强制结束该进程 (**SIGKILL**) 。如果进程不存在，则忽略此条目。

once 当引导 **init** 进入的运行级别匹配条目的 *rstate* 时，启动进程且不等待其结束。当进程结束时，不重启进程。如果引导 **init** 进入一个新的运行级别，但进程仍然从一个先前的运行级别改变中运行着，则进程不会重启。

ondemand 此说明实际上对于 **respawn** 操作同样适用。它在功能上等价于 **respawn**，但已被赋予了一个不同的关键字，从而切断了它与运行级别之间的联系。这只与 *rstate* 字段中描述的 **a** 、 **b** 或 **c** 值一起使用。

powerfail 仅当引导 **init** 接收到一个掉电信号时，与此条目相关的进程才会执行（SIGPWR 请参阅 *signal*(5) ）。

powerwait 仅当引导 **init** 接收到一个掉电信号时，与此条目相关的进程才会执行（SIGPWR 请参阅 *signal*(5) ），并且直到它结束时才会继续处理 **inittab** 。

respawn 如果进程不存在，则启动此进程；不等待其结束（继续扫描 **inittab** 文件）。当进程结束时，重启此进程。如果进程当前已存在，则不做任何事情并继续扫描 **inittab** 文件。

sysinit 这种类型的条目在引导 **init** 试图访问控制台之前执行。此条目应该仅用于初始化设备，且引导 **init** 可能会尝试从这些设备上获取运行级别信息。在继续前执行和等待这些条目。

wait 当引导 **init** 进入的运行级别匹配条目的 *rstate* 时，启动进程并等待其结束。当引导 **init** 在同一运行级别时，任何随后的 **inittab** 文件读取操作会导致引导 **init** 忽略此条目。

process 这是一个要执行的 **sh** 命令。整个 **process** 字段前面使用 **exec** 作为前缀，并作为“ **sh -c 'exec** *command***'**”传递给一个派生的 **sh** 。由于这个原因，任何可以合法地跟在 **exec** 之后的 **sh** 语法都可以出现在 *process* 字段中。可以使用 **; #***comment* 语法插入注释。

警告

强烈推荐使用 4 字符 *id* 。许多 PTY 服务器使用 PTY 名称的最后两个字符作为一个 *id* 。如果一个 PTY 服务器选择的 *id* 和 **inittab** 文件中使用的 ID 相冲突，则可能损坏 **/etc/utmp** 文件。一个损坏的 **/etc/utmp** 文件可能导致命令（例如 **who** ）报告不准确的信息。

文件

/etc/inittab 引导 **init** 所调度的进程文件。

另请参阅

sh(1)、 getty(1M)、 exec(2)、 open(2)、 signal(5)。

名称

inode_vxfs - VxFS 文件系统 i 节点的格式

概要

#include <sys/types.h>

#include <sys/fs/vx_inode.h>

说明

VxFS i 节点长度通常为 256 字节，但也可为 512 字节。可以使用 **mkfs** 指定 i 节点大小。

i 节点条目具有以下格式：

i_mode 文件模式和类型。

i_nlink 文件链接的数目。

i_uid i 节点所有者。

i_gid i 节点组。

i_size 文件大小，以字节为单位。分配 8 字节。

i_atime 最后访问时间，以 **struct timeval** 格式表示。

i_mtime 最后修改时间，以 **struct timeval** 格式表示。

i_ctime i 节点最后更改时间，以 **struct timeval** 格式表示。

i_aflags 下列标志控制文件的分配和扩展：

VX_AF_IBAD 如果设置该项， i 节点无效。运行 **fsck** 时清除。

VX_AF_NOEXTEND 如果设置该项，超过当前保留时间后，文件无法扩展。可以使用 **VX_SETEXT** ioctl 增加保留时间，但文件不会自动扩展。

VX_AF_NOGROW 如果设置该项，超过当前保留时间后，文件无法扩展。通常设置该标志，因为在扩展文件时会出现 I/O 错误。在截断或运行 **setext** 时清除。

VX_AF_ALIGN 如果设置该项，文件必须以固定的大小和对齐方式分配到盘区。如果无法找到在 *i_fixextsize* 边界对齐的 *i_fixextsize* 块的盘区，分配失败。相对于分配单元的开头对齐。

i_orgtype 映射类型。指示如何解释 i 节点映射区域。当前支持 4 种映射类型：

IORG_NONE 不使用映射区域。 **IORG_NONE** 用于无关联数据存储的文件。因为不需要盘区或直接数据，所以不使用映射区域。例如，特殊块和字符文件使用此组织类型。

IORG_EXT4 映射区域包括含 32 位盘区块地址和大小的数组。

IORG_IMMED 映射区域自身为数据块。此映射作为直接 i 节点数据被引用。

IORG_TYPED 映射区域包括类型盘区结构。

i_eopflags 扩展 i 节点操作标志区域。

i_eopdata 扩展 i 节点操作数据区域。

i_ftarea 此字段为并集。内容由文件类型确定。

设备支持下列字段：

i_rdev 块或字符专用设备的设备号。

目录支持下列字段：

i_dotdot 如果 i 节点为目录，则该字段表示 i 节点号的父目录。替换标准“..”（点点）条目，在第一目录块中。 VxFS 没有显式“.”(dot) 和“..”(dot dot) 条目。

常规文件支持下列字段：

i_reserve 保留数据块号，作为文件专用（预分配）。可以使用 ioctl 请求预分配。请参阅 *vxfsio*(7) 。

i_fixextsize i 节点具有固定盘区大小时设置。缺省值为可变盘区大小分配策略。固定盘区大小可以使用 ioctl 指定。请参阅 *vxfsio*(7) 。

结构文件支持下列字段：

i_matchino （仅适用于版本 2 及更高版本的磁盘布局）。“匹配”i 节点的 i 节点号。对于复制的文件，此项为副本的 i 节点。对于盘区映射重新组织文件，此项为重新组织文件的 i 节点。

i_fsetindex （仅适用于版本 2 及更高版本的磁盘布局）。与 i 节点关联的文件集索引。

i_blocks 分配给文件的块号码，包括分配给间接地址盘区的块。

i_gen 生成号。序号，当 i 节点被释放和重新分配时增加。此项为诸如 NFS 之类的无状态服务器提供“句柄”。

i_vversion i 节点元数据的修改次数。此字段为 64 位数字。

ic_org 映射区域。此字段为并集，基于 **i_orgtype** 的值和文件系统类型。

对于 VxFS **IORG_IMMED** 组织类型，使用下列结构：

i_immed 直接 i 节点数据区域，长度为 **NIMMED_N** （当前为 96 字节）（请参阅 **fs_immedlen** ）。任何长度小于或等于 96 字节的目录或符号链接直接存储在 i 节点中。

对于 VxFS **IORG_EXT4** 组织类型，使用下列结构：

i_ies 间接盘区大小。文件中间接数据盘区的大小，以块为单位。

i_ie 间接地址盘区的数组。有 **NIADDR** 间接地址盘区。间接地址盘区长度为 8192 字节。每个间接地址盘区可以最多包括 2048 个盘区地址。

第一个间接地址盘区用于单个间接引用。使用单个间接引用，间接地址盘区中的每个条目指示数据盘区起始块编号。

第二个间接地址盘区是双重间接地址盘区。使用双重间接引用，间接地址盘区中的每个条目指示单个间接地址盘区的起始块编号。

i_dext 结构中包含直接盘区地址和大小的数组。支持最大 **NDADDR_N** 直接盘区。由于具有可变长度盘区策略，每个直接盘区可以有不同大小。每个结构包含如下元素：

i_de 直接盘区地址。

i_des 直接盘区大小。

i_iattrino （仅适用于版本 2 及更高版本的磁盘布局）。间接属性 i 节点。在包含间接属性引用的属性文件集中标识 i 节点。

保留 i 节点的剩余字节用以扩展属性记录。格式如下：

length 属性记录的长度。如果该项不是 4 个字节的倍数，那么通过向上舍入到 4 个字节长度的边界查找下一个属性记录的开始。

format 属性记录剩余部分的数据布局格式。每个属性包括一个用于标识属性管理域的类、一个标识管理域中属性的子类和数据。有效记录格式为：

ATTR_EXTIMMED
扩展直接数据区域，这样大于 96 字节的文件可以直接存储在 i 节点中。

ATTR_IMMED
属性直接存储在 i 节点中。此记录剩余的字段为：

class 属性的类。

subclass 属性的子类。

data 属性数据。

ATTR_DIRECT
当属性太大而不能直接存储在 i 节点中时，每个属性存储在自己的文件中， **ATTR_DIRECT** 根据属性存储的相应文件，列举每个属性及其 i 节点号。列表中的条目号由记录长度确定。每个条目中的字段为：

class 属性的类。

subclass 属性的子类。

length 属性数据的长度。此项允许属性操作检查属性的长度，而无需读取属性 i 节点。

inumber 文件的 i 节点号包含属性数据。 i 节点是属性文件集的一部分。

i 节点中的属性记录以一条格式为零的记录终止（用于兼容文件系统，所有 i 节点最后 80 个字节设置为 NULL）。

另请参阅

setext(1M)、 stat(2)、 fs_vxfs(4)、 vxfsio(7)。

名称

ioconfig - ioconfig 条目格式

概要

#include <sys/ioparams.h>

说明

ioconfig 文件用于保留系统在重新引导之间的 IO 配置信息。它包含两种类型的信息：

- 动态分配的主设备编号到驱动程序的映射。
- 实例编号到硬件路径的映射。

引导时，该文件被读取，并将信息保存到 **io_tree** 内核数据结构中。安装时， **ioconfig** 文件由 **insf** 创建；当添加或删除设备时，由 **insf** 、 **rmsf** 和 **ioscan** 对其进行修改（请参阅 *insf*(1M) 、 *rmsf*(1M) 和 *ioscan*(1M) ）。当系统没有运行时， **ioconfig** 文件的唯一用途是维护配置信息。当系统运行时，直接对内核 **io_tree** 结构进行所有访问，尽管更改内核结构的所有工具必须使 **ioconfig** 保持一致。

系统将维护 **ioconfig** 的两个副本： **/etc/ioconfig** 和 **/stand/ioconfig** 。第二个副本放置在 **/stand** 中，因为在引导时不保证 NFS 无盘客户端有一个可靠的 **/etc** 目录。

ioconfig 文件以 **ioconfig** 幻数开头。

```
#define  IOCONFIG_MAGIC     0x21224941     /* magic number */
```

幻数后面是一个 **ioconfig_record** 结构数组，该数组逻辑上组成一个树结构，定义各层软件模块和管理程序的连接，设备类和每个元素的硬件地址，以及与每个叶子节点相关的逻辑单元。树根是数组元素 0。

每个 **ioconfig_record** 都包含在 **<sys/ioparams.h>** 中定义的下列各字段：

```
#define IOCONFIG_FILE   "/etc/ioconfig"
#define MAX_NAME_LEN        16

union ioconfig_record {
    char      rec_name[MAX_NAME_LEN];          /* record type */
    ioconfig_t  ioc;
    dyn_major_t dm;

} ioconfig_record;
```

每个元素的定义如下：

rec_name 每个记录的第一个条目必须是一个字符串，用于确定记录类型。缺省记录是 **ioconfig_t** 。如果字符串以一个 _ （下划线）字符开头，则它是一种变型。

ioc 这是 ioconfig 文件的缺省记录条目。 **rec_name** 禁止以 _ （下划线）字符开头，以便与其他记录类型相区分。

ioconfig_t 包含下列元素。

```
typedef struct ioconfig {
        char        name[MAX_NAME_LEN];    /* Node name: disc4, scsi_disk */
        char        class[MAX_NAME_LEN];   /* Node class: disk, tape, etc */
        hw_path_t   hw_path;          /* Path to this node      */
        int         instance;         /* Instance number within class*/

} ioconfig_t;
```

dm 此记录存储关于动态指定给驱动程序的主设备编号的信息。它用于允许主设备编号赋值在引导之间持续存在。 **rec_name** 必须以 _ （下划线）字符开头，以便与其他记录类型相区分。

dyn_major_t 包含下列元素。

```
typedef struct dyn_major {
        char    rec_name[MAX_NAME_LEN];        /* Must be  _DYN_MAJOR" */"
        char    name[MAX_NAME_LEN];     /* Driver name      */
        int     c_major;
        int     b_major;
        int     reserved;

} dyn_major_t;
```

作者

ioconfig 由 HP 开发。

文件

/etc/ioconfig

/stand/ioconfig

另请参阅

ioscan(1M)、 ioinit(1M)、 insf(1M)、 rmsf(1M)、 magic(4)。

名称

issue - 发出标识文件

说明

文件 **/etc/issue** 包含 *issue* 或作为登录提示输出的项目标识。这是一个 ASCII 文件，此文件首先会通过 **getty** 程序读取，然后被写入任意从 **inittab** 文件衍生或重新衍生的终端。

文件

/etc/issue

另请参阅

getty(1M)、login(1)。

名称

krb5.conf - Kerberos 配置文件

说明

配置文件 **krb5.conf** 包含 Kerberos V5 库所需的信息。这包括描述缺省 Kerberos 领域的信息以及已知领域的 Kerberos 密钥分发中心的位置。

krb5.conf 是 INI 格式的文件。各节用方括号 **[ ]** 分界。每节内都存在一些关系，在这些关系中可以指定标记具有某些特定值。标记还可以包含小节，而小节可以包含更多的关系和小节。可以为一个标记指定多个值。下面给出 **krb5.conf** 使用的 INI 格式示例：

```
[section1]
    tag1 = value_a
    tag1 = value_b
    tag2 = value_c

[section 2]

    tag3 = {
        subtag1 = subtag_value_a
        subtag1 = subtag_value_b
        subtag2 = subtag_value_c
    }
    tag4 = {
        subtag1 = subtag_value_d
        subtag2 = subtag_value_e
    }
```

krb5.conf 文件一般使用以下节。下面详细地解释这些节：

[libdefaults] 包含 Kerberos V5 库所使用的各个缺省值。

[appdefaults] Kerberos V5 应用程序使用的缺省值。

[login] 包含 Kerberos V5 登录程序 **login.krb5** 所使用的缺省值（注释：Kerberized 登录程序并不是随同产品一起发布的）。

[realms] 包含 Kerberos 领域名称，这些名称描述应在何处查找特定领域的 Kerberos 服务器和其他的特定领域相关信息。

[domain_realm] 包含子域名和域名与 Kerberos 领域名称之间的映射关系。该节由程序用来根据主机的完全限定域名确定主机应位于哪个领域。

[logging] 包含一些用于确定 Kerberos 实体如何进行日志记录的关系。

[capaths] 包含与非层次结构跨领域一起使用的验证路径。客户使用本节中的条目确定跨领域验证中可能使用的中间领域 。终端服务使用本节来检查信任中间领域的转换字段。

libdefaults 节

[libdefaults] 节定义了下列关系：

default_keytab_name 这个关系指定 telnetd 和 rlogind 等应用服务器所使用的缺省 keytab 名称。缺省是 **/etc/krb5.keytab** 。它以前的缺省值是 **/etc/v5srvtab** 。

default_realm 这个关系标识客户主机的 Kerberos 活动中使用的缺省领域。

default_tgs_enctypes 这个关系标识应由密钥分发中心返回的会话密钥加密类型支持列表。支持列表应使用逗号或空格分隔。

default_tkt_enctypes 这个关系以相同的格式标识客户请求的会话密钥加密类型支持列表。

clockskew 本关系设置在假定 Kerberos 消息无效之前库所能够接受的最大允许时钟偏移量，以秒为单位。缺省值是 300 秒或 5 分钟。

kdc_timesync 如果关系的值是非零值，库将计算系统时钟与密钥分发中心返回的时间差。这是为了修正不精确的系统时钟。修正因子只能被 Kerberos 库使用。

kdc_req_checksum_type

有些 DCE 安全服务器不支持该版本 Kerberos 缺省使用的 **CKSUMTYPE_RSA_MD5** ，使用这个关系为了与这些服务器保持兼容。 **CKSUMTYPE_RSA_MD4** 使用值 2 代替。它适用于 DCE 1.1 及更早的版本。

ap_req_checksum_type 这个关系允许您设置 **KRB_AP_REQ** 消息验证程序所使用的校验和类型。该类型的缺省值为 **CKSUMTYPE_RSA_MD5** 。为了与使用 DCE Kerberos 库的应用程序保持兼容，采用值 2 就可以使用 **CKSUMTYPE_RSA_MD4** 。这适用于 DCE 1.1 以及更早的版本。

safe_checksum_type 这个关系允许您设置 **KRB_SAFE** 消息所使用的加密校验和类型。该类型的缺省值为 **CKSUMTYPE_RSA_MD5_DES** 。为了与使用 DCE Kerberos 库的应用程序保持兼容，采用值 3 就可以使用 **CKSUMTYPE_RSA_MD4_DES** 。这适用于 DCE 1.1 以及更早的版本。

ccache_type DCE 客户端上的系统使用这个关系来指定 **kinit** 创建的缓存类型或接收已转发凭证的时间。 DCE 和 Kerberos 共享缓存，但是有些版本的 DCE 不支持该版本的 Kerberos 创建的缺省缓存。DCE 1.0.3a 系统使用值 1， DCE 1.1 系统使用值 2。

appdefaults 节

[appdefaults] 节中的每个标记都命名一个 Kerberos V5 应用程序。标记的值是一个含有关系的小节，这些关系定义了应用的缺省行为。例如：

```
[appdefaults]
  kinit = {
```

```
    forwardable = true
}
```

可以在应用的联机帮助页中查找到应用的可配置选项列表。 **[realms]** 中指定的选项将覆盖该节中指定的应用缺省选项。

login 节

[login] 节用于配置 Kerberos V5 登录程序 **login.krb5** 的行为。

realms 节

文件的 **[realms]** 节中的每个标记命名一个 Kerberos 领域。标记的值是一个小节，小节中的关系定义特定领域的属性。例如：

```
[realms]
  ATHENA.MIT.EDU = {
      kdc = KERBEROS.MIT.EDU
      kdc = KERBEROS-1.MIT.EDU:750
      kdc = KERBEROS-2.MIT.EDU:88
      admin_server = KERBEROS.MIT.EDU
      default_domain = MIT.EDU
      v4_instance_convert = {
          mit = mit.edu
          lithium = lithium.lcs.mit.edu
      }
  }
```

对于每个领域，可以在领域的小节中指定下列标记：

kdc 这个关系的值是一个主机名，领域的密钥分发中心运行在该主机上。可以在主机名后追加一个可选的端口号（并在端口号前面加上一个冒号）。

admin_server 这个关系标识运行管理服务器的主机。它通常是主 Kerberos 服务器。

default_domain 这个关系标识领域中各主机的缺省域。将（不包含域名的） V4 主体名称转换成（包含域名的） V5 主体名称时，需要使用它。

v4_instance_convert 这个子节允许管理员使用缺省域映射规则来配置例外。它包含 V4 实例（标记名称），这些 V4 实例应被转换为与 Kerberos V5 主体名称中的第二部分相似的特定主机名（标记值）。

domain_realm 节

[domain_realm] 节用于将主机名转换为该主机所提供服务的 Kerberose 领域名。

标记名可以是主机名，也可以是域名，域名用一个句点字符前缀（“.”）表示。关系的值是特定主机或域的 Kerberos 领域名称。主机名和域名应小写。

如果不进行转换，那么将认为主机的领域就是转换为大写的主机名的域部分。例如，下面的 **[domain_realm]** 节：

```
[domain_realm]
    .mit.edu = ATHENA.MIT.EDU
    mit.edu = ATHENA.MIT.EDU
    dodo.mit.edu = SMS_TEST.MIT.EDU
    .ucsc.edu = CATS.UCSC.EDU
```

将 **dodo.mit.edu** 映射为 **SMS_TEST.MIT.EDU** 领域。 **MIT.EDU** 域中的所有其他主机映射到 **ATHENA.MIT.EDU** 领域， **UCSC.EDU** 域中的所有主机映射到 **CATS.UCSC.EDU** 领域。缺省规则将 **ucbvax.berkeley.edu** 映射到 **BERKELEY.EDU** 领域。 **sage.lcs.mit.edu** 映射到 **LCS.MIT.EDU** 领域。

logging 节

[logging] 节说明特定实体如何进行日志记录。本节中指定的关系将一个或多个值分配给实体名。

当前使用下列实体：

kdc 这些条目规定密钥分发中心如何进行日志记录。

admin_server 这些条目规定管理服务器如何进行日志记录。

default 这些条目规定在没有明确规范的情况下应如何进行日志记录。

各个值的形式如下：

FILE=*filename*
FILE:*filename* 这个值导致在特定文件中存储实体日志消息。如果使用 **=** 形式，则文件被覆盖。否则将向文件追加日志记录。

STDERR 这个值导致实体日志消息被输出到标准错误流。

CONSOLE 如果系统支持，这个值将导致实体日志消息输出至控制台。

DEVICE=*devicename* 导致实体日志消息输出至特定设备。

SYSLOG[:*severity***[:***facility***]]** 导致实体日志消息输出至系统日志。

severity 参数指定系统日志消息的缺省级别。它可以是 **syslog()** 调用（请参阅 *syslog*(3C) 联机帮助页）所支持的下列级别中的任何一个级别。支持的参数为

LOG_ALERT LOG_CRIT LOG_DEBUG LOG_ERR

LOG_EMERG LOG_INFO LOG_NOTICE LOG_WARNING

例如，若要指定 **LOG_CRIT** 级别，可以使用 **CRIT** 作为 **severity** 参数。 **LOG_** 前缀被删除。

facility 参数指定记录日志消息的工具。它可能是 **syslog()** 调用（请参阅 *syslog*(3C) ）支持的以下工具中的任何一个。支持的参数包括： **LOG_KERN** 、 **LOG_USER** 、 **LOG_MAIL** 、 **LOG_DAEMON** 、 **LOG_AUTH** 、 **LOG_LPR** 、 **LOG_NEWS** 、 **LOG_UUCP** 、 **LOG_CRON** 和 **LOG_LOCAL0** 至 **LOG_LOCAL7** 。

如是没有指定 **severity** ，缺省值为 **ERR** 。如果没有指定 **facility** ，缺省值为 **AUTH** 。

在下面的例子中， **LOG_DAEMON** 工具使用缺省的 **LOG_INFO** 级别将密钥分发中心的日志消息输出到控制台。管理服务器发出的日志消息将被追加到 **/var/adm/kadmin.log** 文件中并发送至 **/dev/tty04** 设备。

```
[logging]
    kdc = CONSOLE
    kdc = SYSLOG:INFO:DAEMON
    admin_server = FILE:/var/adm/kadmin.log
    admin_server = DEVICE=/dev/tty04
```

capaths 节

跨领域验证一般是分层组织的。这种分层结构以领域的名称为基础。因此对领域名的选择和跨领域验证参与者的选择进行了约束。也可以使用非层次结构的组织，但是它需要使用一个数据库来构建各领域间的验证路径。本节定义了这个数据库。

客户端使用本节查找它的领域与服务器领域间的验证路径。通过检查接收到的凭证的转换字段，服务器可以使用本节来验证客户端所使用的验证路径。

每个参与领域都有一个标记名称。每个标记都为领域设置了一个子标记。子标记的值是一个中间领域，这个中间领域可能参与跨领域验证。如果有多个中间领域，子标记可能重复出现。值“.”意味着两个领域直接共享密钥，不允许中间领域的参与。

表可能有 **n**2** 个条目，但是只需给出客户端和服务器所需要的条目。客户的本地领域需要一个标记，还需要一些子标记来表示验证服务器的所有领域。服务器需要用标记表示它所服务的客户端的所有领域。

例如， **ANL.GOV** 、 **PNL.GOV** 和 **NERSC.GOV** 希望将 **ES.NET** 领域用作中间领域。 **ANL** 有一个 **TEST.ANL.GOV** 子领域，这个子领域与 **NERSC.GOV** 进行验证，但是不与 **PNL.GOV.** 进行检证。 **ANL.GOV** 系统的 **[capaths]** 节应是这样的：

```
[capaths]
    ANL.GOV = {
        TEST.ANL.GOV = .
        PNL.GOV = ES.NET
        NERSC.GOV = ES.NET
        ES.NET = .
    }
    TEST.ANL.GOV = {
        ANL.GOV = .
    }
    PNL.GOV = {
        ANL.GOV = ES.NET
    }
    NERSC.GOV = {
```

```
        ANL.GOV = ES.NET
    }
    ES.NET = {
        ANL.GOV = .
    }
```

NERSC.GOV 系统所使用配置文件的 **[capaths]** 节应是这样的:

```
[capaths]
    NERSC.GOV = {
        ANL.GOV = ES.NET
        TEST.ANL.GOV = ES.NET
        TEST.ANL.GOV = ANL.GOV
        PNL.GOV = ES.NET
        ES.NET = .
    }
    ANL.GOV = {
        NERSC.GOV = ES.NET
    }
    PNL.GOV = {
        NERSC.GOV = ES.NET
    }
    ES.NET = {
        NERSC.GOV = .
    }
    TEST.ANL.GOV = {
        NERSC.GOV = ANL.GOV
        NERSC.GOV = ES.NET
    }

    }
```

在上面的 例子中，顺序并不重要，多次使用同一个子标记名称时除外。客户端使用它来确定路径（对于服务器而言它并不重要，因为没有排序转换字节）。

如果没有此节，或者如果客户端或服务器不能查到客户端/服务器路径，则假定为标准分层结构。

DCE 目前不支持该功能。DCE 安全服务器可以与 Kerberized 客户端和服务器一起使用，但是 DCE 1.1 以前的版本不能在转换字段中使用，因此使用时应十分谨慎。

文件

/etc/krb5.conf
Kerberos 配置文件。

/usr/contrib/krb5/sample/krb5.conf.sample
本产品附带的示例 Kerberos 配置文件。

作者

krb5.conf 由麻省理工学院开发。

另请参阅

syslog(3C)、 kerberos(5)。

名称

libgss - GSSAPI 共享库（一般安全服务应用程序编程界面）

概要

#include<gssapi.h>

/usr/lib/libgss.sl

说明

libgss 为共享库，按照 RFC 2743 它包含所有 GSSAPI，并按 RFC 2744 中定义的 C 语言接口实现，一般安全服务 API：C 绑定。

GSSAPI 为独立于各基础安全机制的应用程序提供安全服务。服务包括验证、完整性和/或保密服务。GSSAPI 使用称为安全内容的数据结构在两个对等端之间提供安全通信。GSSAPI 调用者负责在对等端之间传送令牌。GSSAPI 独立于基础通信协议。

建立安全连接的应用程序是内容初始用户或简单初始用户。接受安全连接的应用程序是内容接受用户或简单接受用户。

使用 GSSAPI C 绑定接口的开发者可以链接应用程序和 **libgss.sl** 。基础安全机制在运行时可以由称为 **/etc/gss/mech** 的配置文件指定，库将从配置文件中指定的路径动态加载相应机制指定的共享库（例如，Kerberos 中的 **libgssapi_krb5.sl** ）。

/etc/gss/mech 文件格式如下:

first column	包括支持 GSSAPI 的后端安全机制名。
second column	包括对象标识符 (OID)。
third column	包括共享库的名称，该库实现 GSSAPI 后端安全机制（后端库为 32 位时必须置于 **/usr/lib/gss** 路径，为 64 位版本时必须置于 **/usr/lib/pa20_64/gss** 路径）

机制文件的缺省路径 (**/etc/gss/mech**) 可以使用 **GSSAPI_MECH_CONF** 环境变量更改。

Example /etc/gss/mech file

```
# Mechanism Name     Object Identifier     Shared Library
#
krb5_mech            1.2.840.113554.1.2.2  libgssapi_krb5.sl
```

除该配置文件之外，还有两个配置文件 **/etc/gss/qop** 和 **/etc/gss/gsscred.conf** ，可用于关联 **libgss.sl** 。

对于基础安全机制， **/etc/gss/qop** 文件包含每个基于 GSSAPI 保护质量 (QOP) 的信息。 **/etc/gss/qop** 文件格式如下:

first column	指定 QOP 字符串名称。
second column	包含 QOP 值（32 位整数）。

third column 包含安全机制名称。

Example /etc/gss/qop file

```
# QOP string           QOP Value    Mechanism Name
#
GSS_KRB5_INTEG_C_QOP_DES_MD5 0           kerberos_v5
```

/etc/gss/gsscred.conf 是选择基础机制的配置文件，用于存储 **gsscred** 表。 **gsscred** 表用于存储安全主体和 UNIX uid 之间的映射。支持的 **gsscred** 后端机制仅仅是普通文件。因此，要执行库的后续操作，必须在 **/etc/gss/gsscred.conf** 中指定“文件”条目。

Example /etc/gss/gsscred.conf file

```
# gsscred configuration file
#
# Valid gsscred backend mechanisms are:
#    files
#
files
```

使用 GSSAPI 框架:

使用 GSSAPI 框架通信的应用程序需要经过如下主要阶段:

1. 通信应用程序需要获取凭证集以向其他应用程序证明其身份。应用程序的凭证证明其全局的身份。

2. 应用程序使用其凭证建立连接安全内容。此信息用于为每个消息提供安全服务，类似于完整性和保密性。

   建立安全内容过程如下：一个应用程序（客户端）启动内容设置。另一个应用程序（服务器）在交换零个或多个令牌后接受它。交换数取决于基础安全机制。

3. 对每个消息服务应用保密性和完整性。应用程序传送希望受到保护的消息时将调用 GSSAPI 例行程序（ gss_get_mic 或 gss_wrap 调用）以进行保护。传送应用程序指定相应安全内容并发送至接收应用程序。接收方传递数据至相应解码例行程序（gss_verify_mic 或 gss_unwrap 调用）以分别删除保护并验证数据。

4. 通信会话结束后，对等端应用程序调用 GSSAPI 例行程序以删除安全内容。

   已在 **libgss.sl** 库中实现了 API 名称，下文将给出每个名称的简短说明。有关使用这些 API 的详细说明，请参考相应联机帮助页。

凭证处理 API

凭证用于验证一个应用程序与另一个应用程序之间的标识。GSSAPI 假定应用程序的凭证已经到位。下列 GSS-API 用于查询和处理它们。

gss_acquire_cred : 允许应用程序通过名称获取原有凭证的句柄。

gss_release_cred : 通知 GSSAPI， 进程不再需要指定的凭证。当所有进程已经发布了凭证后，它将被删除。

gss_inquire_cred : 获取有关凭证的信息。

gss_inquire_cred_by_mech : 获取每个有关凭证的机制信息。

gss_add_cred: 通过递增来构建凭证。

内容处理 API

对于建立连接安全内容并用于安全服务的应用程序，需要下列 GSSAPI。

gss_init_sec_context ： 使用对等端应用程序启动安全内容。

gss_accept_sec_context ： 接受由对等端应用程序启动的安全内容。

gss_delete_sec_context ： 放弃安全内容（释放内容结构内存）。

gss_inquire_context ： 获取有关安全内容信息。

gss_process_context_token ： 处理从对等端应用程序来的令牌环。

gss_export_sec_context ： 传输安全内容至另一进程。

gss_import_sec_context ： 导入传输内容。

gss_context_time ： 确定内容保持有效的时限。

gss_wrap_size_limit ： 确定内容上的 **gss_wrap** 令牌大小限制。

每个消息操作 API

要像执行数据完整性和数据私密性服务那样执行每个消息操作，可以使用下列 GSSAPI：

gss_get_mic ： 用于计算加密消息的消息完整性代码 (MIC) （完整性服务）。

gss_verify_mic ： 用于验证消息完整性。

gss_wrap ： 用于为私密性服务加密消息。

gss_unwrap ： 用于为私密性服务解密消息。

名称处理 API

名称标识主体。名称以可打印格式表示（显示给其他应用程序），或以内部格式（规范格式）标识，用于 API 并对应用程序不透明。下列为用于操作名称的 API：

gss_import_name ： 将邻近字符串名称转换为内部格式。

gss_display_name ： 将内部格式名称转换为文本。

gss_compare_name ： 对比两个内部格式名称。

gss_release_name ： 放弃内部格式名称。

gss_inquire_names_for_mech ： 列举指定机制支持的名称。

gss_inquire_mechs_for_name ： 列举支持指定名称类型的机制。

gss_canonicalize_name： 将内部名称转换为机制指定的机制名称 (MN)。

gss_export_name： 将 MN 转换为平面名称表示，适用于逐位对比。

gss_duplicate_name： 创建内部名称的副本。

其他操作 **API**

除了上文提到的 API 类别外，还有用于其他操作的 API，如创建和删除机制 OID 列表，以文本格式显示 GSSAPI 错误状态代码， 为缓冲区释放内存等，如下所示：如

gss_add_oid_set_member： 为 OID 集添加对象标识符。

gss_release_oid_set： 释放 OID 集的内存。

gss_create_empty_oid_set： 在没有 OID 时创建 OID 集。

gss_test_oid_set_member： 确定 OID 是否为集的成员。

gss_release_buffer： 释放缓冲区内存。

gss_release_oid： 释放 OID 结构内存。

gss_release_name： 释放名称类型内存。

gss_display_status： 转换 GSSAPI 状态代码为文本。

gss_indicate_mechs： 确定可用基础安全机制。

作者

libgss.sl 由 Sun Microsystems, Inc. 开发。

另请参阅

gssapi(5)。

DCE-GSSAPI 的联机帮助页包括在 DCE-CoreTools 产品中。要参阅该联机帮助页，请将 **/opt/dce/share/man** 添加至 **MANPATH** 。

名称

lif - 逻辑交换格式说明

说明

LIF （逻辑交换格式）是 HP 标准的海量存储格式，可以用于在不同的 HP 计算机系统之间进行文件交换。 LIF 卷包含一个标题（标识其为 LIF 卷）以及定义卷的内容（即文件）的一个目录。初始化卷时，目录的大小是固定的（请参阅 *lifinit*(1) ）并设定可以在卷上创建的文件的数量上限。

HP-UX 包含实用程序的一个集合（称为 *lif*∗(1) ），可以用于：

- 初始化 LIF 卷（即，创建一个标题和空目录），
- 将文件复制到 LIF 卷和从 LIF 复制，
- 列出 LIF 卷的内容，
- 删除 LIF 文件，
- 重命名 LIF 文件。

lif∗(1) 实用程序是 HP-UX 中仅有的 LIF 卷的内部结构已知的实用程序。对于其余的 HP-UX， LIF 卷仅为包含某些未指定数据的文件。术语 **LIF** 卷不应当与 **HP-UX** 表示形式的文件系统卷和可挂接的卷相混淆。

HP-UX 上的 LIF 实用程序当前支持三种文件类型， ASCII (1)、 BINARY (-2) 和 BIN (-23951)。

三种复制模式与这些文件类型相关联：

ASCII 如果复制模式为 ASCII 而且一个 HP-UX 文件正被复制到 LIF 卷，则实用程序去除结尾的 LF （换行符）字符，并在每条记录前插入两字节的记录长度。这些记录就会被写入 LIF 格式的介质。当将 LIF ASCII 文件复制到 HP-UX 时，两字节的记录长度会被去除，而且结尾的 LF 会被追加。这些记录会被写入目的地。在这种复制模式中，将保留文件长度。此复制模式的缺省文件类型为 ASCII (1)。

BINARY 如果复制模式为 **BINARY** 而且正在将一个 HP-UX 文件复制到 LIF 卷中，实用程序仅仅在每 1K 字节记录前插入两字节的记录长度。结尾的部分的块有一个反映此块中字节数量的数字。记录的内容上没有解释。这些记录会被写入 LIF 格式的介质中。当以 **BINARY** 复制模式将一个 LIF 文件复制到 HP-UX 文件时，记录长度会被去除而且记录的内容会直接被写入目的地。在复制模式中，二进制文件的长度将保留。此复制模式缺省的文件类型为 **BINARY** (-2)。

RAW 如果复制模式为 RAW，而且一个 HP-UX 文件正被复制到 LIF 卷中，则实用程序仅仅将原始数据复制到目的地。不为 256 字节的整数倍的文件大小会用 null （空）填充至下一个更高的倍数。因此， 文件大小不保留。当以 RAW 模式将一个 LIF 文件复制到 HP-UX 文件中时，信息直接被复制，而源的内容上不做任何解释。此复制模式的缺省文件类型为 **BIN** (-23951)。

LIF 卷可以在任意 HP-UX 文件（常规的磁盘文件，或设备专用文件）上创建，支持通过 **lseek()** （请参阅 *lseek*(2)）随机访问。 请勿在使用前挂接专用文件 *lif*∗(1) *utilities* 。有关详细信息，请参阅 *lifinit*(1) 。在 LIF 卷中，单个文件通过 1 到 10 个字符的文件名标识。文件名可以包括大写的字母数字字符（A 到 Z，0 到 9）以及下划线 (_)。 LIF 文件名的第一个字符必须为字母（A 到 Z） 。 *lif*∗(1) 实用程序接受任意文件名（包括其他系统上生成的非法文件名） ，但是只能创建合法的名称。这表示名称中包含小写字母的文件只能被读取，不能创建。

通过连接 LIF 卷的 HP-UX 路径名， LIF 文件名被指定给 *lif*∗(1) 实用程序，后接 LIF 文件名，两者之间通过冒号 (**:**) 分隔。例如：

/dev/fd.0:ABC	指定通过 HP-UX 设备专用文件 **/dev/fd.0** 访问的 LIF 文件 **ABC** 。
myfile:ABC	指定 HP-UX 中的磁盘文件 **myfile** LIF 文件 **ABC** 。

注意文件命名惯例仅在用作 *lif*∗(1) 实用程序的参数时适用，而且在 HP-UX 操作系统中不构成有效的路径命名。

请勿在使用时挂接专用文件 *lif*∗(1) *utilities* 。

另请参阅

lifcp(1)、 lifinit(1)、 lifls(1)、 lifrename(1)、 lifrm(1)。

名称

localedef - 语言环境定义文件的格式和语义

说明

这是对语言环境定义的语法和含义的说明，所提供的语言环境定义作为用来创建语言环境的 **localedef** 命令的输入（请参阅 *localedef*(1M)）。

下面列出了可由 **localedef** 识别的类别标记、关键字和后续表达式。除了 **copy** 关键字以及 **LC_COLLATE** 说明中标注的其他例外情况之外，任一类别的关键字的顺序是不相关的。（请注意，按照惯例，类别标记由大写字符组成，而关键字由小写字符组成）。

类别标记和关键字

下列关键字不属于任何类别，而且应当出现在语言环境定义文件的开头：

comment_char
单个字符，指示要解释为语言环境定义文件中注释行的开始的字符。该字符应当位于注释行的第一列中。缺省的 *comment_char* 是 **#** 。将忽略第一列中包含 *comment_char* 的所有行。

escape_char
单个字符，指示要解释为脚本中转义符的字符。缺省的 *escape_char* 是 **** 。 *escape_char* 用于对 localedef 元字符进行转义以去除特殊含义，它采用字符常量十进制、八进制和十六进制格式。如果 *escape_char* 是一行的最后一个字符（位于换行符之前），它还用于将该行延续到下一行。

下列关键字可用于任意类别：

copy 一个字符串，它命名系统中另一个有效的语言环境。这会导致将语言环境中的类别创建为已命名语言环境中同一个类别的副本。由于 **copy** 关键字定义整个类别，因此，如果使用它，它必须是该类别中的唯一关键字。

可识别下列六个类别：

LC_CTYPE：
该类别定义字符分类、大小写转换和其他字符属性。可识别下列预定义的字符分类：

upper 归为大写字母的字符代码。在 **cntrl** 、 **digit** 、 **punct** 或 **space** 分类中指定的字符，不能在该类别中指定。

lower 归为小写字母的字符代码。适用于 **upper** 类别的限制同样适用于该分类。

digit 归为数字的字符代码。只能按数值的连续升序指定十个字符。不能在此处指定备用数字。

space 归为空格的字符代码。为 **upper** 、 **lower** 、 **alpha** 、 **digit** 、 **graph** 或 **xdigit** 类别指定的字符，不能包括在该分类中。

punct 归为标点字符的字符代码。不能指定包括在 **upper** 、 **lower** 、 **alpha** 、 **digit** 、 **cntrl** 、 **xdigit** 或 **space** 类别中的字符。

cntrl 归为控制字符的字符代码。不能在此包含那些包括在 **upper** 、 **lower** 、 **alpha** 、 **digit** 、 **punct** 、 **graph** 、 **print** 或 **xdigit** 中的字符。

blank 归为空白字符的字符代码。将自动包括 <space> 和 <tab> 字符。

xdigit 归为十六进制数字的字符代码。只能指定为 **digit** 类定义的字符，并在其后添加一组或多组六个字符，每一组都按升序排列。

alpha 归为字母的字符代码。不能指定归为 **cntrl** 、 **digit** 、 **punct** 或 **space** 的字符。在该类中将自动包括指定为 **upper** 和 **lower** 类的字符。

print 归为可打印字符的字符代码。将自动包括 <space> 字符以及为 **upper** 、 **lower** 、 **alpha** 、 **digit** 、 **xdigit** 和 **punct** 类指定的字符。不能指定 **cntrl** 类别中的字符。

graph 归为可打印字符的字符代码， <space> 字符除外。在所有其他方面，该分类与 **print** 类别相似。

下面是两个特殊的分类，它们分别用于指定双 **bytes** 中第一个和第二个有效字节。请注意，它们是字节分类，而不是字符分类；因此，不能采用其他分类所使用的方式将它们与 *iswctype* 接口一起使用（请参阅 *wctype*(3C) ） 。

first 双字节字符中的第一个有效字节。

second 双字节字符中的第二个有效字节。

字符大小写转换定义:

toupper 小写字符转换为大写字符的关系。

tolower 大写字符转换为小写字符的关系。

其他字符属性和分类:

alt_punct 映射到 ASCII 等效字符串“b!"#$%&'()*+,-./:;<=>?@[\]^_'{}~”的字符串，其中 b 是空白字符（一个 *langinfo*(5) 项目） 。

charclass 将一个或多个语言环境特定的字符类名定义为由分号分隔的 *strings* 。随后，可以在 **LC_CTYPE** 定义中定义每个命名字符类。字符类名中的第一个字符必须是字母，而且类名不能与任何预定义的分类（如 **space** 、 **letter** 和 **cntrl** ） 相匹配。

direction 指示文本方向的字符串操作数（一个 *langinfo*(5) 项目） 。字符串操作数“1”指示文本方向为从右到左。

context 指示字符环境分析的字符串操作数。字符串“1”指示需要进行阿拉伯语环境分析。

LC_COLLATE:

LC_COLLATE 类别为语言环境中的排序元素（单字符排序元素和多字符排序元素）的相对顺序提供排序序列定义。下列关键字属于该类别，它们应当出现在类别标记 **LC_COLLATE** 和 **END LC_COLLATE** 之间。前两个关键字可以按任意顺序显示，但必须位于 **order_start** 关键字之前。前两个关键字可以指定

任意多次。

collating-element <*symbol*> from *string*

定义一个多字符排序元素 *symbol* ，该元素由 *string* 中的字符组成。 *String* 限制为两个字符。

collating-symbol <*symbol*>

使 *symbol* 成为可用于在排序序列中定义位置的排序符号。 *Symbol* 不代表任何实际字符。

order_start 表示排序序列的开头。这些指令对字符串排序有影响。

位于 **order_start** 关键字之后、 **order_end** 关键字之前的行包含排序元素条目，一行一个条目。

操作数可以有选择地出现在所定义规则的 **order_start** 关键字后面，以便使用多权重方案对字符串进行比较（如果未指定操作数，则假设使用单个 **forward** 操作数）。可能的操作数包括:

forward 指定比较运算将从字符串的开头朝着字符串的末尾进行。

backward 指定比较运算将从字符串的末尾朝着字符串的开头进行。

order_end 标记排序元素条目列表的末尾。

LC_MONETARY:

LC_MONETARY 类别定义用来格式化货币数值信息的规则和符号。下列关键字属于该类别，它们应当出现在类别标记 **LC_MONETARY** 和 **END LC_MONETARY** 之间:

int_curr_symbol

该操作数是一个用于指定国际货币符号的四字符字符串。

currency_symbol

该操作数是一个用作本地货币符号的字符串。

mon_decimal_point

该操作数是一个字符串，其中包含用作十进制定界符（小数分隔符）的符号。

mon_thousands_sep

该操作数是一个字符串，其中包含用作十进制定界符左侧多组数字的分隔符的符号。

mon_grouping 该操作数是一个用分号分隔的整数列表。最初的整数定义紧邻十进制定界符前面的组的大小，后面的整数定义前面的组。如果最后一个整数不是 -1，则前一个组（如果有）的大小将重复用于其余的数字。如果最后一个整数是 -1，则将不再进一步执行分组。

positive_sign 该操作数是一个用来指示非负货币数量的字符串。

negative_sign 该操作数是一个用来指示负货币数量的字符串。

int_frac_digits 该操作数是一个整数，代表使用 **int_curr_symbol** 的格式化货币值中所使用的小数位数。

frac_digits 该操作数是一个整数，代表使用 **currency_symbol** 的格式化货币值中所使用的小数位数。

p_cs_precedes 该操作数是一个整数，如果将它设置为 1，则指示 **currency_symbol** 或 **int_curr_symbol** 位于货币数量前面；如果将它设置为 0，则指示该符号位于货币数量后面。

p_sep_by_space 该操作数是一个整数，如果将它设置为 1，则指示用空格将 **currency_symbol** 或 **int_curr_symbol** 与值分开；否则会将它设置为 0。

n_cs_precedes 该操作数是一个整数，如果将它设置为 1，则指示 **currency_symbol** 或 **int_curr_symbol** 位于负货币数量前面；如果将它设置为 0，则该符号将位于负值后面。

n_sep_by_space 该操作数是一个整数，如果将它设置为 1，则指示用空格将 **currency_symbol** 或 **int_curr_symbol** 与负货币值分开；否则会将它设置为 0。

p_sign_posn 该操作数是一个整数，其设置指示非负货币数量的 **positive_sign** 的位置。可能值包括：

0 用括号将数量和 **currency_symbol** 或 **int_curr_symbol** 括起来。

1 符号字符串位于数量和 **currency_symbol** 或 **int_curr_symbol** 前面。

2 符号字符串位于数量和 **currency_symbol** 或 **int_curr_symbol** 后面。

3 符号字符串位于 **currency_symbol** 或 **int_curr_symbol** 前面。

4 符号字符串位于 **currency_symbol** 或 **int_curr_symbol** 后面。

n_sign_posn 该操作数是一个整数，其设置与 **p_sign_posn** 的设置相似，但它适用于负货币数量。

LC_NUMERIC:

LC_NUMERIC 类别定义用来格式化非货币数值信息的规则和符号。下列关键字属于该类别，它们应当出现在类别标记 **LC_NUMERIC** 和 **END LC_NUMERIC** 之间：

decimal_point 该操作数是一个字符串，其中包含用作数值、非货币格式化数量中十进制定界符（小数分隔符）的符号。不能省略此关键字，而且不能将其设置为空字符串。

thousands_sep 该操作数是一个字符串，其中包含用作十进制定界符左侧多组数字的分隔符的符号。

grouping 该操作数是一个用分号分隔的整数列表。最初的整数定义紧邻十进制定界符前面的组的大小，后面的整数定义前面的组。如果最后一个整数不是 -1，则前一个组（如果有）的大小将重复用于其余的数字。如果最后一个整数是 -1，则将不再进一步执行

分组。

alt_digit 映射到 ASCII 等效字符串“0123456789b+-.,eE”的字符串，其中 b 是空白字符（一个 *langinfo*(5) 项目）。 **alt_digit** 关键字是 HP 对 **localedef** POSIX 标准的扩展，它与在 POSIX 标准中定义的 **alt_digits** 具有不同的含义。

LC_TIME:

LC_TIME 类别定义用来生成语言环境特定的格式化日期字符串的规则。下列必需的关键字属于该类别，它们应当出现在类别标记 **LC_TIME** 和 **END LC_TIME** 之间:

abday 七个用分号分隔的字符串，为一周中的每一天提供简称，从星期日开始。

day 七个用分号分隔的字符串，为一周中的每一天提供全称，从星期日开始。

abmon 十二个用分号分隔的字符串，为月份提供简称，从一月开始。

mon 十二个用分号分隔的字符串，为月份提供全称，从一月开始。

d_t_fmt 该操作数是用来定义适当的日期和时间表示形式的字符串。

d_fmt 该操作数是用来定义适当的日期表示形式的字符串。

t_fmt 该操作数是用来定义适当的时间表示形式的字符串。

am_pm 该操作数是两个用分号分隔的字符串，它们提供 **AM** 和 **PM** 的表示形式。

t_fmt_ampm 该操作数是一个字符串，它用带有 **am_pm** 的 12 小时制格式定义适当的时间表示形式。

era 该操作数是一个用分号分隔的字符串列表。每个字符串为一个语言环境定义纪元或帝号的名称和日期。每个字符串应当遵循以下格式:

direction **:** *offset* **:** *start_date* **:** *end_date* **:** *name* **:** *format*

其中:

direction **+** 或 **-** 字符。 **+** 字符指示时间轴应当为：在从起始日期朝着结束日期移动时，年份沿着正方向计数。 **-** 字符指示时间轴应当为：在从起始日期朝着结束日期移动时，年份沿着负方向计数。

offset **[SHRT_MIN,SHRT_MAX]** 范围中的数字，指示纪元的起始年份。

start_date 格式为 *yyyy***/***mm***/***dd* 的日期，其中 *yyyy* 、 *mm* 和 *dd* 分别是纪元的起始年份、月份和日期。 公元 0 年之前的年份以负数表示。例如，从 公元前 100 年 3 月 5 日开始的纪元将表示为 **3-100/3/5** 。支持 **[SHRT_MIN+1,SHRT_MAX-1]** 范围中的年份。

end_date 纪元的结束日期，其形式与上面的 *start_date* 或者两个特殊值 **-*** 或 **+*** 之一相同。值 **-*** 指示将纪元的结束日期扩展到开始时间，

而 **+*** 指示将纪元的结束日期扩展到结束时间。可以按年代顺序将结束日期排列在纪元的开始日期之前或之后。例如，公元和公元前基督纪元的表达式将为：

+:0:0000/01/01:+*:A.D.:%o %N

+:1:-0001/12/31:-*:B.B.:%o %N

name 一个字符串，代表用于替换 **date** 和 **strftime()** （请参阅 *date*(1) 和 *strftime*(3C) ）的 **%N** 指令的纪元名称。

format 一个字符串，用于格式化 *date*(1) 和 *strftime*(3C) 的 **%E** 指令。该字符串通常是 **%o** 和 **%N** 指令的函数。如果未指定 *format* ，则会将为 **LC_TIME** 类别关键字 **era_d_fmt** （请参阅下文）指定的字符串用作缺省值。

era_d_fmt 该操作数是一个用来定义纪元表示法日期格式的字符串。

era_t_fmt 该操作数是一个用来定义纪元表示法时间格式的字符串。

era_d_t_fmt 该操作数是一个用来定义纪元表示法日期和时间格式的字符串。

alt_digits 该操作数是一个用分号分隔的字符串列表。第一个字符串是与零相对应的替代符号，第二个字符串是与一相对应的替代符号，以此类推。请注意，如果已在同一个语言环境中指定了 HP-UX 专有的 **alt_digit** 关键字，则这两个关键字的前十个符号应当相同。

除了上面的关键字之外，还可以识别 HP-UX 专有的下列关键字（提供这些关键字是为了实现向后兼容，不建议使用它们）： **year_unit** 、 **mon_unit** 、 **day_unit** 、 **rour_unit** 、 **min_unit** 、 **sec_unit** 。

LC_MESSAGES:

LC_MESSAGES 类别定义肯定响应和否定响应的格式和值。下列关键字属于该类别，它们应当出现在类别标记 **LC_MESSAGES** 和 **END LC_MESSAGES** 之间：

yesexpr 该字符串操作数是扩展的正则表达式，它将可接受的肯定响应与“是/否”查询相匹配。

noexpr 该字符串操作数是扩展的正则表达式，它将可接受的否定响应与“是/否”查询相匹配。

yesstr 该字符串操作数标识“是/否”问题的肯定响应。此关键字现已过时，应当使用 **yesexpr** 。

nostr 该字符串操作数标识“是/否”问题的否定响应。此关键字现已过时，应当使用 **noexpr** 。

关键字操作数

关键字操作数由字符代码常量和符号、字符串及元字符组成。合法表达式的类型包括： **character lists** 、 **string lists** 、 **integer lists** 、 **shift** 、 **collating element entries** 、 **regular expression** 、 **character constants** 和 **string** ：

character lists

character list 操作数由用分号分隔的单字符代码常量或符号名称组成，或者由一个字符代码范围组成，该范围包括一个常量或符号名称，其后是一个省略号，省略号后面是另一个常量或符号名称。省略号前面的常量的代码值必须小于省略号后面的常量的值。范围代表一组连续的字符代码。如果该列表多于一行，则必须在每一行的末尾使用转义符并将其用作续行符。使用未在随附的 charmap 文件（请参阅 *charmap*(4) ）中定义的任何符号名称是错误的。

string lists **string list** 操作数由用分号分隔的字符串组成。如果多于一行，则必须使用转义符进行续行。

string **string** 操作数由零个或多个用双引号 (") 括起的字符序列组成。在一个字符串中，必须在转义符前面加上双引号字符。还可使用下列转义符序列：

\n 换行符

\t 横向制表符

\b 退格符

\r 回车符

\f 换页符

**** 反斜杠

\’ 单引号

****_ddd_ 位模式

转义符 *ddd* 由后跟用来指定所需字符值的 1 位、2 位或 3 位八进制数的转义符组成（有关其他可能的位模式规范，请参阅下面的字符常量）。此外，将忽略转义符 (\\) 和紧跟在换行符之后的内容。

尽管在阐释时使用的是反斜杠 (\\)，但是仍可以用 **escape_char** 关键字来替换另一个转义符。

character constants

常量用操作数来代表字符代码。它们可在下列形式中使用：

decimal constants 依次后跟 **’d’** 和最多三个十进制数的转义符。

octal constants 后跟最多三个八进制数的转义符。

hexadecimal constants 依次后跟 **’x’** 和两个十六进制数的转义符。

character constants 单个字符（例如，A），具有计算机上字符集中字符的数值。

symbolic names

括在 < 和 > 之间的字符串是符号名称。建议完全用符号名称的形式编写 **localedef** 输入文件，以便利用用户定义的或系统提供的 charmap 文件。这有助于在不同的编码字符集之间移植 **localedef** 输入文件（请参阅 *charmap*(4)）。

可以用 **collating-element** 和 **collating-symbol** 关键字在语言环境定义文件中定义符号名称。这些名称不是字符常量。如果这样一个内部定义的符号名称与 charmap 文件中定义的符号名称发生冲突，则会出现错误。

integer lists

Integer list 操作数由一个或多个用分号分隔的十进制数组成。

shift

Shift 操作数跟在 **toupper** 和 **tolower** 关键字后面，它们必须由两个用左括号和右括号括起来并用逗号分隔的字符代码常量组成。每个这样的字符对之间都用分号分隔。对于 **tolower** ，第一个常量代表大写字符，第二个常量代表相应的小写字符。对于 **toupper** ，第一个常量代表小写字符，第二个常量代表相应的大写字符。

collating element entry

order_start 关键字后跟排序元素条目，每行一个条目，这些条目按照排序位置的升序排列。排序元素条目的格式如下：

collation_element[*weight*[*;weight*]]

collation_element 可以是一个字符、一个代表字符或排序元素的用方括号括起来的排序符号、特殊符号 **UNDEFINED** 或省略号 (**...**) 。

字符代表其自身；排序符号可以是由 charmap 文件解释的字符的符号名称、由 **collating-element** 关键字定义的多字符排序元素，或者由 **collating-symbol keyword** 定义的排序符号。

特殊符号 **UNDEFINED** 指定不是由排序元素条目明确定义的任何字符的排序位置。例如，如果要从排序序列中省略某组字符，而且这些字符刚好排在所有已定义字符的后面，则可以在 **order_start** 关键字前面定义一个排序符号：

collating-symbol <HIGH>

则排序元素条目列表中的某个位置为：

UNDEFINED <HIGH>

请注意，没有第二个权重。这表示在第二次排序时，所有的字符都按其编码值进行排序。

省略号被解释为字符的列表，这些字符的编码值大于上一行字符的值，小于下一行字符的值。因为省略号绑定到字符的编码值，所以它本质上是不可移植的。如果使用省略号，除非指定了 **-c** 选项，否则会发出一个警告，而且不会生成任何输出。

weight 操作数提供有关在第一次和后续排序时将如何对排序元素进行排序的信息。 *Weight* 可以是两字符字符串、特殊符号 **IGNORE** ，或者为 *collating_element* 指定的任何形式（ **UNDEFINED** 除外）的排序元素。如果没有 *weights* ，则会严格按照字符在列表中的位置对字符进行排序。如果只给出了一个 *weight* ，则在第二次排序时，将按照字符在列表中的相对位置对字符进行排序。

等效类由一系列排序元素条目定义，所有这些条目在第一个 *weight* 位置有相同的字符或符号。例如，在许多语言环境中，在第一次排序时，所有形式的字符“A”都按照相同的方式进行排序。在排序元素条目中这可表示为：

’A’ ’A’;’A’ # first element of equivalence class
’a’ ’A’;’a’ # next element of class

二对一排序元素由在 **order_start** 关键字前面定义的 *collating-elements* 来指定。例如，在西班牙语中，二对一排序元素 **CH** 将在 **order_start** 关键字前面定义为

collating element <CH> from "CH"

然后，它将在排序元素条目中用作 **<CH>** 。

一对二排序元素可通过在其中一个 *weight* 位置中设置两字符字符串来定义。例如，如果字符 **’X’** 的排序顺序与字符对“AE”相同，则排序元素条目将为：

’X’ "AE";’X’

无关字符由特殊符号 **IGNORE** 定义。例如，在第一次排序时，短线字符 **’-’** 可以是无关字符。排序元素条目如下：

’-’ IGNORE;’-’

由 **collating-symbol** 关键字定义的符号可用来指示给定字符在排序顺序上比某个位置高或低。例如，如果在第一次排序时，要使编码值小于 **’0’** 值的所有字符的排序顺序比所有其他字符低，并且在第二次排序时采用相对顺序，请在 **order_start** 关键字前面定义一个排序符号：

collating-symbol <LOW>

则前两个排序元素条目为：

... <LOW>;...
’0’ ’0’;’0’

这还阐释了省略号在指示范围时的用法。第一个省略号被解释为“编码字符集内值小于‘0’的所有字符”；第二个省略号表示第一次以相对顺序排序时所定义的范围中的所有字符。

regular expression

regular expression 操作数遵循扩展的正则表达式规范，如 *regexp*(5) 中所述。

元字符

元字符是指在操作数中对 *localedef* 具有特殊含义的字符。为了对这些字符的特殊含义进行转义，请用单引号括起它们或者在它们前面添加一个转义符。 *localedef* 元字符包括：

< 指示符号名称的开头。

> 指示符号名称的末尾。

(指示后跟 **toupper** 和 **tolower** 关键字的字符切换对的开头。

) 指示字符切换对的末尾。

, 用于分隔字符切换对的字符。

" 用于将字符串括起来。

; 用作列表操作数中的分隔符。

escape character
: 用于对其他元字符及自身的特殊含义进行转义。它在缺省情况下为反斜杠 (\)，但是可以用 **escape_char** 关键字重新定义。

注释

注释是以注释字符开头的行。注释字符在缺省情况下为井字符 (#)，但是可以用 **comment_char** 关键字重新定义。注释和空白行将被忽略。

分隔符

分隔符包括空白字符和制表符。可以使用任意数量的分隔符来分隔包含 *localedef* 脚本的关键字、元字符、常量和字符串，但 < 和 > 之间的所有字符除外，即使它们是 <空白>，也将被视为符号名称的一部分。

举例

有关语言环境说明文件的示例，请参阅 **/usr/lib/nls/loc/src** 下的文件。这些文件用于创建随 HP-UX 提供的各种语言环境。

名称

lvmpvg - LVM 物理卷组信息文件

概要

/etc/lvmpvg

说明

lvmpvg 是为系统中所有物理卷组保存卷组信息的 ASCII 文件。该信息以分层格式保存。

首先，它以一个卷组开始，在该卷组下可以存在多个物理卷组。在每个物理卷组之下，可以指定一个物理卷列表。出现在此文件中的每个卷组中必须至少有一个物理卷组。物理卷组名在相应的卷组中必须唯一，虽然允许在不同的卷组中使用相同的物理卷组名。此文件中的卷组可以与系统中的一样多。

管理员可以编辑此文件来创建和扩展物理卷组，而不是使用 **vgcreate** 和 **vgextend** 命令。然而，必须小心以保证包含在该文件中的所有物理卷已经通过之前使用 **vgcreate** 或 **vgextend**进行了定义。

lvmpvg 文件格式具有如下结构。 **VG** 和 **PVG** 是分别引入 *volume group* 和 *physical volume group* 的名称的关键字。 此文件中不允许有注释 。

> **VG** *vg_name*
> **PVG** *pvg_name*
> *pv_path*
> . . .
> **PVG** *pvg_name*
> *pv_path*
> . . .
> **VG** *vg_name*
> **PVG** *pvg_name*
> *pv_path*
> . . .

变量定义如下:

pv_path 卷组中物理卷的块设备路径名。

pvg_name 物理卷组名。它在卷组中必须唯一。

vg_name 卷组的路径名。

举例

下面的例子显示了一个包含两个卷组的 **lvmpvg** 文件：第一个包含两个物理卷组，每个组中都定义了两个物理卷；第二个包含三个物理卷组，每个组中定义了一个物理卷。

> **VG /dev/vg00**
> **PVG PVG0**
> **/dev/dsk/c2t0d0**

/dev/dsk/c2t1d0
PVG PVG1
/dev/dsk/c3t0d0
/dev/dsk/c3t1d0
VG /dev/vg01
PVG PVG0
/dev/dsk/c4t0d0
PVG PVG1
/dev/dsk/c5t0d0
PVG PVG2
/dev/dsk/c6t0d0

另请参阅

vgcreate(1M)、 vgextend(1M)、 vgreduce(1M)、 vgremove(1M)。

名称

magic - HP-UX 实现的幻数

概要

#include <magic.h>

说明

magic.h 文件在一个文件中本地化了关于 HP-UX “幻数”的所有信息，从而帮助统一了对幻数的处理。此文件指定幻数在一个文件中的位置（总是在文件开头）及幻数的结构：

```
struct magic_number {
   unsigned short    system_id;
   unsigned short    file_type;
};
typedef     struct    magic_number MAGIC;
```

magic.h 包括所有运行 HP-UX 的 HP 系统 ID的定义，以及与所有实现相同的文件类型。可能有附加的实现相关的文件类型。预定义的文件类型为：

```
/* for object code files */
   #define RELOC_MAGIC     0x106      /* relocatable only */
   #define EXEC_MAGIC      0x107      /* normal executable */
   #define SHARE_MAGIC     0x108      /* shared executable */
   #define DEMAND_MAGIC    0x10B      /* demand-load executable */
   #define LISP_MAGIC      0x10C      /* compiled Lisp */
   #define DL_MAGIC        0x10D      /* dynamic load library */
   #define SHL_MAGIC       0x10E      /* shared library */
   #define HPE_MAGIC       0x150      /* HPE boot image */
```

system_id 的值定义在 *model*(4) 中。

警告

通过 **cpio** 管理的文件使用与 **<magic.h>** 兼容的不同格式的幻数。

另请参阅

ar(1)、 ld(1)、 a.out(4)、 ar(4)、 model(4)。

名称

mnttab - 挂接的文件系统表

概要

#include <mntent.h>

说明

mnttab 驻留在目录 **/etc** 中，并包含通过 **mount** 命令挂接的设备表（请参阅 *mount*(1M)）。对于每个挂接的文件系统，文件包含一行信息，结构上等同于通过 *fstab*(4) 描述的 **/etc/fstab** 。

有一定数量的此种格式的行：

special_file_name dir type opts freq passno mount_time

组成条目相似于：

/dev/dsk/c0d0s0 / hfs rw 0 1 537851723

/etc/mnttab 通过使用 **getmntent()** 的程序访问（请参阅 *getmntent*(3X)）。永远不应对其手动编辑，也不应该使用 **setmnt** 来在 **/etc/mnttab** 中创建条目（请参阅 *setmnt*(1M)）。

mount_time 包含文件系统使用 **mount** 挂接的时间。此值是从 Epoch (00:00:00 国际标准时间，1 月，1970 年）开始的秒数，请参阅 *time*(2) 。

文件系统挂接或取消挂接时， **mount** 和 **umount** 更新 **mnttab** 。如果文件对于由 HP-UX 内核内部维护的已挂接的文件系统表已过期，则 **syncer** 将重写 **mnttab** 文件（请参阅 *syncer*(1M)）。

警告

仅当此表作为程序用以返回关于挂接的文件系统信息的一种方式时才提供。

/etc/mnttab 永不应被手动编辑。任何对于 **/etc/mnttab** 的手动更改在没有警告的情况下，都将被 **syncer** 、 **mount** 和 **umount** 覆盖。

作者

mnttab 由 HP、加州大学伯克利分校和 Sun Microsystems, Inc. 联合开发。

文件

/etc/mnttab

另请参阅

mount(1M)、 getmntent(3X)、 fstab(4)。

名称

model - HP-UX 计算机标识

概要

#include <model.h>

说明

由于硬件的不同， HP-UX 实现之间不可避免地存在差别。在这样的差别存在的地方，可以使用条件编译或其他定义来隔离不同。在 **<model.h>** 头文件中收集了解析这些差别的标志和类型定义，该头文件中包含定义各种 HP-UX 实现的常量。

例如，头文件 **model.h** 包含在 **<sys/magic.h>** 中定义了值的常量：

```
#define HP_S_500 HP9000_ID
#define HP_S_200 HP98x6_ID
#define HP_S_300 CPU_HP_MC68020
#define HP_S_800 CPU_PA_RISC1_0
#define HP_S_700 CPU_PA_RISC1_1
```

其他此类常量将根据需要在 HP-UX 扩展到其他计算机时在后续版本中添加。

此外， **model.h** 具有定义预处理器常量 **MYSYS** 的语句表示所需编译的特定实现。 **MYSYS** 总是等于上面的一个常量。

如果该文件包含基于实现或基于体系结构的功能，则条件编译可以用于适应在多于一个 HP-UX 实现上执行。例如，代码段：

```
#if MYSYS==HP_S_400
        <statements>
#endif
```

仅当系统处理器是 HP 9000 系列 400 计算机时，才编译 if 语句之后的语句。

model.h 也包含几个预定义类型的类型定义来加强某些类型代码和文件的可移植性。

执行类型定义声明被头文件 **inttypes.h** 替换

int8，**u_int8**	有符号及无符号 8 位整型。
int16，**u_int16**	有符号及无符号 16 位整型。
int32，**u_int32**	有符号及无符号 32 位整型。
machptr，**u_machptr**	足够保存指针的有符号及无符号整型。

要使端口号使用 **inttypes.h** 而不是 **model.h** ，它们的相应定义包含于 **intypes.h** 中

int8_t，**uint8_t**	有符号及无符号 8 位整型。
int16_t，**uint16_t**	有符号及无符号 16 位整型。

int32_t, **uint32_t**	有符号及无符号 32 位整型。
intptr_t, **uintptr_T**	足够保存指针的有符号及无符号整型。

定义了某些 C 预处理器条件编译变量来帮助基于实现的代码。请参阅 *cpp*(1)。

另请参阅

cc(1)、 cpp(1)、 inttypes(5)、 magic(4)。

名称

named.conf - NameDaemon 的配置文件

概要

/etc/named.conf

说明

BIND 9 配置与 BIND 8.x 大体相似。但是，它包含一些新的配置区域，如视图。虽然需要查看更加复杂的配置，以检查是否可以使用 BIND 9.2 中实现的新功能来更有效地实现这些配置，但是 BIND 8.x 配置文件应该处理 BIND 9.2 中的较少改动。 BIND 4.9.7 配置文件可以通过使用 Shell 脚本 **/usr/bin/named-bootconf.sh** 转换为 BIND 9.2 格式。

配置文件元素

在 BIND 9.2 配置文件文档中使用下列配置元素：

acl_name *address_match_list* 的名称，由 **acl** 语句定义。

address_match_list
一个或多个 *ip_addr, ip_prefix, key_id,* 或 *acl_name* 元素的列表。

domain_name 将用作 DNS 名称的引用字符串，例如 **my.test.domain** 。

dotted_decimal 仅由点号 (**.**) 分隔的一个或多个整数，取值范围是 0 到 255，如 123、45.67 或 89.123.45.67。

ip4_addr 正好包含四个以 *dotted_decimal* 形式表示的元素的 IPv4 地址。

ip6_addr IPv6 地址，如 fe80::200:f8ff:fe01:9742。

ip_addr *ip4_addr* 或 *ip6_addr* 。

ip_port IP 端口号。它限制在 0 到 65535 的范围， 1024 以下的值通常限制为根拥有的进程。某些情况下，星号 (*****) 字符可用作占位符来选择随机高编号端口。

ip_prefix 指定为 *ip_addr* 的 IP 网络，后接斜线 (**/**) 以及网络掩码中的位数。可能会省略 *ip_addr* 中的尾随零。例如， 127/8 是网络掩码为 255.0.0.0 的网络 127.0.0.0， 1.2.3.0/28 是网络掩码为 255.255.255.240 的网络 1.2.3.0。

key_id 表示共享密钥名称的 *domain_name* ，用于确保事务安全。

key_list 一个或多个 *key_ids* 的列表，用分号分隔且用分号结尾。

number 非负 32 位无符号整数（即 0 和 4294967295 之间的数字，且包括这两个数）。它的可接受值可能会由其使用环境进行进一步限制。

path_name 将用作路径名的引用字符串，例如 **zones/master/my.test.domain** 。

size_spec 数字、单词“unlimited”或单词“default”。 unlimited 的 *size_spec* 请求无限的使用量，或者可用的最大使用量。 default 的 *size_spec* 使用在启动服务器时强加的限制。数字可以选择性地后接缩放因子：“K”或“k”表示千字节，“M”或“m”表示兆字节，“G”或“g”表示千兆字

节，它们分别放大 1024、1024*1024 和 1024*1024*1024。该值必须可表示为 64 位无符号整数（0 到 18446744073709551615，且包括这两个数）。使用“unlimited”是设置很大数字的最佳方式。

yes_or_no “yes”（是）或“no”（否）。也接受单词“true”和“false”以及数字 1 和 0。

dialup_option *yes*、*no*、*notify*、*notify-passive*、*refresh* 或 *passive* 之一。在区域中使用时，*notify-passive*、*refresh* 和 *passive* 限制在从属和存根区域。

地址匹配列表语法

address_match_list3D address_match_list_element **;**
[*address_match_list_element*; **...**]
address_match_list_element3D [**!**] (*ip_address* [**/***length*] |
key key_id | *acl_name* | { *address_match_list* })

地址匹配列表定义和用法

地址匹配列表主要用于确定各种服务器操作的访问控制。它们也用于定义查询其他名称服务器的优先级，并设置 **named** 将用来监听查询的地址。构成地址匹配列表的元素可以是下面的任何对象：

- IP 地址（IPv4 或 IPv6）
- IP 前缀（在“/”表示法中）
- 密钥 ID，由 **key** 语句定义
- 先前用 **acl** 语句定义的地址匹配列表的名称
- 括号中的嵌套地址匹配列表

元素可以用前导感叹号 (**!**) 来求反。 **any** 、 **none** 、 **localhost** 和 **localnets** 的匹配列表名称是预定义的。有关这些匹配列表名称的详细信息，请参考 **acl** 语句一节。由于安全密钥可用于验证访问，而不考虑主机或网络地址，因此添加 key 子句可使该句法元素的名称变得用词不当。但是，术语“地址匹配列表”仍在使用。

给定 IP 地址或前缀与地址匹配列表进行比较时，将按顺序遍历该列表，直到元素匹配。匹配的解释取决于该列表是否用于访问控制，定义 **listen-on** 端口，以及该元素是否已求反。当用作访问控制列表时，非求反匹配允许访问而求反匹配拒绝访问。如果没有匹配项，则拒绝访问。

子句 **allow-notify** 、 **allow-query** 、 **allow-transfer** 、 **allow-update** 和 **blackhole** ，它们可以使用地址匹配列表在选项和（或）区域中指定。同样， **listen-on** 选项可使服务器不接受对任何不匹配列表的计算机地址的查询。

由于该算法的首先匹配特性，定义列表中其他元素子集的元素应该位于边界元素之前，而不管是否求反。例如，在 **1.2.3/24; ! 1.2.3.13;** 中，1.2.3.13 元素毫无用处，因为该算法将匹配从 1.2.3.13 到 1.2.3/24 元素的任何查找。使用 **! 1.2.3.13; 1.2.3/24** ，可通过让求反阻挡 1.2.3.13 来解决该问题，但是会遗漏其他所有 1.2.3.* 主机。

注释语法

BIND 9.2 配置文件中的注释可以用 C、C++ 或 Shell/Perl 结构来编写。

警告：与区域文件不同，不能使用分号 (**;**) 字符在 BIND 9.2 配置文件中开始注释。分号指示配置语句的结束。

配置文件语法

BIND 9.2 配置文件由语句和注释组成。语句以分号结尾。只有语句和注释才能在不带括号的情况下使用。许多语句包含子语句块，子语句用分号终止。支持下列语句：

acl 定义命名的 IP 地址匹配列表，用于访问控制和其他目的。

controls 声明要由 **rndc** 实用程序使用的控制通道。

include 包含一个文件。

key 指定要用于 TSIG 身份验证和授权的密钥信息。

logging 指定服务器日志记录的内容以及日志消息的发送位置。

options 控制全局服务器配置选项，并设置其他语句的缺省值。

server 逐个服务器地设置特定配置选项。

trusted-keys 定义可信的 DNSSEC 密钥。

view 定义视图。

zone 定义区域。

logging 和 **options** 语句在一次配置操作中只能出现一次。

acl 语句语法

acl *acl-name* **{**
address_match_list
};

acl 语句定义和用法

acl 语句给地址匹配列表分配符号名称。其名称源于地址匹配列表的主要用途：“访问控制列表”(ACL)。请注意，在使用地址匹配列表名称之前，必须使用 **acl** 将其定义；不允许提前引用。下列 ACL 是内置的：

any 匹配所有主机。

none 不匹配任何主机。

localhost 匹配系统上所有网络接口的 IPv4 地址。

localnets 匹配 IPv4 网络上系统具有其接口的任何主机。

localhost 和 **localnets** ACL 目前不支持 IPv6（即 **localhost** 不匹配主机的 IPv6 地址， **localnets** 不匹配主机的连接 IPv6 网络），这是因为缺少确定主机的完整本地 IPv6 地址集的标准方法。

controls 语句语法

controls {
inet (*ip_addr*| *****) [**port** *ip_port*] **allow {** *address_match_list* **}**
keys { *key_list* **};**

```
[ inet ...; ]
};
```

controls 语句定义和用法

controls 语句声明将由系统管理员用来影响本地名称服务器操作的控制通道。这些控制通道由 **rndc** 实用程序用来向名称服务器发送命令，以及从名称服务器检索非 DNS 结果。

inet 控制通道是可供 Internet 访问的 TCP/IP 套接字，它在指定 *ip_addr* 上的指定 *ip_port* 处创建。如果未指定端口，则使用缺省端口 953。 * 无法用于 *ip_port* 。

通过控制通道发出命令的能力受到 **allow** 和 **keys** 子句的限制。将根据 **address_match_list** 中的地址权限来允许连接到控制通道。将忽略 **address_match_list** 的 **key_id** 成员，而根据 **key_list** 对其进行独立的解释。 **key_list** 中的每个 **key_id** 均可用来对通道提供的命令和响应进行身份验证，方法是给服务器和命令客户端之间的每一消息添加数字签名。控制通道的所有命令必须由将采用的指定密钥之一进行签名。

如果不存在 **controls** 语句， **named** 将设置一个缺省控制通道，它在其环回地址 127.0.0.1 及其等效 IPv6 地址 ::1 上监听。在这种情况下，以及在 **controls** 语句存在，但不包含 **keys** 子句时， **named** 将尝试从 **/etc** 中的 **rndc.key** 文件加载命令通道密钥。要创建 **rndc.key** 文件，请运行 **rndc-confgen -a** 。 **rndc.key** 功能已实现，为从 BIND 8 转换系统提供了方便，它在命令通道消息上没有数字签名，因此不包含 **keys** 子句。

由于 **rndc.key** 功能的目的只是允许向后兼容使用 BIND 8 配置文件，所以该功能不具有高度可配置性。由于无法轻易地更改密钥名称和保密内容大小，因此应该用自己的密钥来创建 **rndc.conf** （如果要将其更改）。 **rndc.key** 文件还设置了特定的权限，只有该文件的所有者（ **named** 运行时所使用的用户身份）才能访问它。如果需要更大的灵活性，允许其他用户访问 **rndc** 命令，则需要创建一个 **rndc.conf** ，使其可由包含应具有访问权的用户的组进行读取。

BIND 8 的 UNIX 控制通道类型在 BIND 9.2 中不受支持，并且不会在将来发行版中添加。如果它出现在 BIND 8 配置文件的 **controls** 语句中，则会将其忽略，并记录一条警告。

include 语句语法

include *filename***;**

include 语句定义和用法

include 语句在遇到 **include** 语句的位置点插入指定的文件。 **include** 语句便于通过允许读取或写入某些内容而不允许访问其他内容来管理配置文件。例如，该语句可以包含只能由名称服务器读取的私钥。

key 语句语法

```
key key_id {
  algorithm string;
  secret string;
};
```

key 语句定义和用法

key 语句定义用于 TSIG 的共享私用密钥。 **key** 语句可以在配置文件的顶级或者在 **view** 语句内出现。在顶级 key 语句中定义的密钥可以在所有视图中使用。要在 **controls** 语句中使用的密钥必须在顶级定义。

key_id 也称作密钥名，它是唯一标识密钥的域名。它可以在 *server* 语句中用来通过该密钥给请求签名，或者在地址匹配列表中用来验证传入的请求是否已经使用与该名称、算法或保密内容相匹配的密钥签名。

algorithm_id 是指定安全性和（或）身份验证算法的字符串。“hmac-md5”是当前用 TSIG 身份验证支持的唯一算法。 **secret_string** 是将由该算法使用的基本 64 编码保密字符串。

logging 语句语法

```
logging {
  [ channel channel_name {
    ( file path name
       [ versions ( number | unlimited ) ]
       [ size size spec ]
     | syslog syslog_facility
     | stderr
     | null );
    [ severity (critical | error | warning | notice |
            info | debug [ level ] | dynamic ); ]
    [ print-category yes or no; ]
    [ print-severity yes or no; ]
    [ print-time yes or no; ]
  }; ]
  [ category category_name {
    channel_name ; [ channel_name ; ... ]
  }; ]
  ...
};
```

logging 语句定义和用法

logging 语句配置名称服务器的多种日志记录选项。其 *channel* 短语将输出方法、格式选项和严重性级别与一个名称相关联，该名称可以与 category 短语一起使用来选择如何记录各种消息。

只有一个 **logging** 语句用于定义任意数量的通道和类别。如果不存在 **logging** 语句，日志记录配置将是：

```
logging {
    category "unmatched" { "null"; };
    category "default" { "default_syslog"; "default_debug"; };
};
```

在 BIND 9.2 中，只有当整个配置文件经过分析时，才建立日志记录配置。在 BIND 8 中，它在分析 **logging** 语句后立即建立。服务器启动时，所有与配置文件中的语法错误相关的日志记录消息将进入缺省通道，如果指定了 **-g** 选项，则进入标准错误。

channel 短语

所有日志输出均进入一个或多个用户定义或预定义的 *channels* 。每个 *channel* 定义必须包括一个目标子句，它指定为通道选择的消息是进入文件、特定 syslog 工具、还是标志错误数据流，或者是被忽略。它还可以选择性地限制通道将接受的消息严重性级别（缺省值是 **info** ），以及是否包括生成 named 的时间戳、类别名称和（或）严重性级别（缺省值是根本不包括）。当 **null** 目标子句忽略向通道发送的所有消息时，通道选项将无关。

file 目标子句将通道定向到磁盘文件。它可以包括对文件大小以及文件版本数的限制，并且在每次打开文件保存一次。

如果使用 **versions** 日志文件选项，则 **named** 将通过在打开时重命名来保留该数量的文件备份版本。

例如，如果选择保留文件 **lamers.log** 的三个旧版本，则在将其打开之前：

lamers.log.1 重命名为 **lamers.log.2** ，

lamers.log.0 重命名为 **lamers.log.1** ，且

lamers.log 重命名为 **lamers.log.0** 。

如果不需要限制版本数，请使用 **versions unlimited**; 。如果 **size** 选项与日志文件相关联，则仅在所打开的文件超过指定大小时进行重命名。缺省情况下不保留任何备份版本；只会追加任何现有日志文件。

文件的 **size** 选项用于限制日志增长。如果文件大小超过限制， **named** 将停止写入该文件（除非它有关联的 **versions** 选项）。如果保留了备份版本，则按照上述方式滚动文件，并打开一个新文件。如果没有 **versions** 选项，则在删除日志文件或将日志文件截断到小于最大大小之前，将无法向日志写入更多数据。缺省行为是不限制文件大小。

size 和 **versions** 选项用法示例：

```
channel "an_example_channel" {
   file "example.log" versions 3 size 20m;
   print-time yes;
   print-category yes;
};
```

syslog 目标子句将通道定向到系统日志。其参数是 *syslog*(3C) 联机帮助页中所述的 syslog 工具。 *syslog*(3C) 联机帮助页介绍 **syslog** 将如何处理发送到该工具的消息。如果您的系统使用非常早的 **syslog** 版本，它仅使用 **openlog()** 函数的两个参数，则将忽略 **syslog** 目标子句。

severity 子句的工作方式与 syslog 的“优先级”类似，不同的是如果直接写入文件而不使用 **syslog** ，也可使用优先级。将不为通道选择低于给定严重性级别的消息；而将接受较高严重性级别的消息。如果使用 **syslog** ，则 **syslog.conf** 优先级还将确定最终通过的对象。

例如，如果将 channel 工具和严重性定义为 **daemon** 和 **debug** ，但仅通过 **syslog.conf** 记录 **daemon.warning** ，则将导致丢弃严重性信息和通知的消息。如果这种情况反过来，即 **named** 仅写入具有警告或更高严重性的消息，则 **syslogd** 将输出它从通道接收的所有消息。

stderr 目标子句将通道定向到服务器的标准错误数据流。它应该在服务器作为前台进程运行时（如调试配置时）使用。

服务器处于提示模式时可以提供大量的调试信息。如果服务器的全局调试级别大于零，则调试模式将处于活动状态。全局调试级别通过以后接正整数的 **-d** 标志启动命名服务器或者通过运行 **rndc** 跟踪来设置。通过运行 **rndc notrace** ，可以将全局调试级别设置为零，并关闭调试模式。服务器中的所有调试消息均具有调试级别，较高的调试级别会提供更详细的输出。例如：

```
channel "specific_debug_level" {
   file "foo";
   severity debug 3;
};
```

在上例中，每当服务器处于调试模式，指定特定调试严重性的通道将得到级别 3 或更低级别的调试输出，而不管全局调试级别是什么。具有 **dynamic** 严重性的通道将使用服务器的全局级别来确定要输出的消息。

如果 **print-time** 已打开，则将记录日期和时间。可以为 syslog 通道指定 **print-time** ，但由于 **syslog** 也会输出日期和时间，这通常是无意义的。如果请求 **print-category** ，还将记录消息的类别。最后，如果 **print-severity** 已打开，则将记录消息的严重性级别。 **print-** 选项可以按任意组合使用，它们将始终按以下顺序输出： time、category、severity。在以下示例中，所有三个 **print-** 选项均已打开：

28-Feb-2000 15:05:32.863 general: notice: running

其中有四个预定义通道，它们用于 named 的缺省日志记录，如下所示：

```
channel "default_syslog" {
   syslog daemon;             // send to syslog's daemon
                         // facility
   severity info;            // only send priority info
                         // and higher
};

channel "default_debug" {
   file "named.run";             // write to named.run in
                         // the working directory
                         // Note: stderr is used instead
                         // of "named.run"
                         // if the server is started
                         // with the '-f' option.
   severity dynamic;             // log at the server's
                         // current debug level
};
```

```
channel "default_stderr" {          // writes to stderr
    stderr;
    severity info;                  // only send priority info
                                    // and higher
};

channel "null" {
   null;                            // toss anything sent to
                                    // this channel
};
```

default_debug 通道具有特殊的属性，即它仅在服务器的调试级别是非零值时才生成输出。它通常写入服务器工作目录中的文件 **named.run** 。

出于安全原因，在使用 **-u** 命令行选项时， **named.run** 文件仅在 **named** 更改为新 UID 后创建，并且当 **named** 启动，且在忽略根时仍然运行时，将生成调试输出。如果需要捕获该输出，必须以 **-g** 选项运行服务器，并将标准错误重定向到文件。

通道定义后，将无法重新定义。因此不能直接更改内置通道，但可以通过将类别指向所定义的通道来修改缺省的日志记录。

category 短语

通过预定义的类别，管理员可以微调要记录的消息以及要将这些消息记录到的位置。如果没有为类别指定通道列表，则会将该类别中的日志消息发送到 **default** 类别。如果未指定 **default** 类别，则将使用以下类别：

```
category "default" { "default_syslog"; "default_debug"; };
```

例如，如果要将安全性事件记录到文件中，并且还希望保留缺省的日志记录行为，则需要指定：

```
channel "my_security_channel" {
    file "my_security_file";
    severity info;
};

category "security" {
    "my_security_channel";
    "default_syslog";
    "default_debug";
};
```

要忽略类别中的所有消息，请指定 **null** 通道，如下所示：

```
category "xfer-out" { "null"; };
category "notify" { "null"; };
```

下面是可用类别及其包含的日志信息类型的简短说明。将来的 BIND 发行版中可能会添加更多的类别。

default default 类别为未定义特定配置的类别定义日志记录选项。

general 综合类别。所有未分类的类别均属于该类别。

database 与名称服务器在内部使用来存储区域和缓存数据的数据库相关的消息。

security 批准和拒绝请求。

config 配置文件分析和处理。

resolver DNS 解析，如缓存名称服务器代表客户端执行的递归查找。

xfer-in 服务器正在接收的区域传输。

xfer-out 服务器正在发送的区域传输。

notify NOTIFY 协议

client 客户端请求处理

unmatched **named** 无法确定其所属类或者不存在其匹配视图的类的消息。一个单行摘要也将记录到 client 类别。最好将该类别发送到文件或 stderr，它在缺省情况下将发送到 null 通道。

network 网络操作

update 动态更新

queries 启用查询日志记录

dispatch 将传入的数据包发送到要在其中进行处理的服务器模块。

dnssec DNSSEC 和 TSIG 协议处理。

lame-servers 无效服务器。它们是远程服务器中的错误配置，是 BIND 9 在解析过程中尝试查询这些服务器时发现的。

lwres 语句语法

下面是 **named.conf** 文件中 **lwres** 语句的语法:

```
lwres {
  [ listen-on { ip_addr [port ip_port] ; [ ip_addr [port ip_port] ;
    ... ] }; ]
  [ view view_name; ]
  [ search { domain_name ; [ domain_name ; ... ] }; ]
  [ ndots number; ]
};
```

lwres 语句定义和用法

lwres 语句配置名称服务器，使其也充当轻量级解析服务器。可以使用多个 **lwres** 语句来配置具有不同属性的轻量级解析服务器。 **listen-on** 语句指定轻量级解析守护程序应该接受其上请求的地址和端口的列表。如果未指定端口，则使用端口 921。如果省略该语句，则将接受 127.0.0.1 端口 921 上的请求。

view 语句将轻量级解析守护程序的特定实例绑定到 DNS 命名空间中的一个视图，以便按照构造匹配该视图的常规 DNS 查询的相同方式构建响应。如果省略该语句，则将使用缺省视图，如果没有缺省视图，则将触发错误。

search 语句等效于 **/etc/resolv.conf** 中的 search 语句。它提供将追加到查询中相对名称的域的列表。

ndots 语句等效于 **/etc/resolv.conf** 中的 **ndots** 语句。它指示在追加搜索路径之前应该导致确切匹配查找的相对域名中的最少点号数。

options 语句语法

下面是 **named.conf** 文件中 **options** 语句的语法：

```
options {
   [ version version_string; ]
   [ directory path_name; ]
   [ tkey-domain domainname; ]
   [ tkey-dhkey key_name key_tag; ]
   [ dump-file path_name; ]
   [ pid-file path_name; ]
   [ statistics-file path_name; ]
   [ zone-statistics yes_or_no; ]
   [ auth-nxdomain yes_or_no; ]
   [ dialup dialup_option; ]
   [ minimal-responses yes_or_no; ]
   [ notify yes_or_no | explicit; ]
   [ recursion yes_or_no; ]
   [ forward ( only | first ); ]
   [ forwarders { ip_addr [port ip_port] ;
             [ ip_addr [port ip_port] ; ... ] }; ]
   [ allow-notify { address_match_list }; ]
   [ allow-query { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ allow-recursion { address_match_list }; ]
   [ allow-v6-synthesis { address_match_list }; ]
   [ blackhole { address_match_list }; ]
   [ listen-on [ port ip_port ] { address_match_list }; ]
   [ listen-on-v6 [ port ip_port ] { address_match_list }; ]
   [ query-source [ address ( ip_addr | * ) ]
```

```
                  [ port ( ip_port | * ) ]; ]
[ max-transfer-time-in number; ]
[ max-transfer-time-out number; ]
[ max-transfer-idle-in number; ]
[ max-transfer-idle-out number; ]
[ tcp-clients number; ]
[ recursive-clients number; ]
[ serial-query-rate number; ]
[ transfer-format ( one-answer | many-answers ); ]
[ transfers-in  number; ]
[ transfers-out number; ]
[ transfers-per-ns number; ]
[ transfer-source (ip4_addr | *) [port ip_port] ; ]
[ transfer-source-v6 (ip6_addr | *) [port ip_port] ; ]
[ notify-source (ip4_addr | *) [port ip_port] ; ]
[ notify-source-v6 (ip6_addr | *) [port ip_port] ; ]
[ also-notify { ip_addr [port ip_port] ;
               [ ip_addr [port ip_port] ; ... ] }; ]
[ coresize size_spec ; ]
[ datasize size_spec ; ]
[ files size_spec ; ]
[ stacksize size_spec ; ]
[ cleaning-interval number; ]
[ heartbeat-interval number; ]
[ interface-interval number; ]
[ sortlist { address_match_list }];
[ lame-ttl number; ]
[ max-ncache-ttl number; ]
[ max-cache-ttl number; ]
[ sig-validity-interval number ; ]
[ use-ixfr yes_or_no ; ]
[ provide-ixfr yes_or_no; ]
[ request-ixfr yes_or_no; ]
[ min-refresh-time number ; ]
[ max-refresh-time number ; ]
[ min-retry-time number ; ]
[ max-retry-time number ; ]
[ port ip_port; ]
[ additional-from-auth yes_or_no ; ]
```

```
    [ additional-from-cache yes_or_no ; ]
    [ random-device path_name ; ]
    [ max-cache-size size_spec ; ]
    [ match-mapped-addresses yes_or_no; ]
    [ edns yes_or_no; ]
};
```

options 语句定义和用法

options 语句设置将由 BIND 使用的全局选项。该语句只能在一个配置文件中出现一次。如果发现出现多次，则第一个实例将确定实际使用的选项，并将生成警告。如果没有 **options** 语句，则将使用每个选项均设置为其缺省值的选项块。

version 服务器应该通过类 CHAOS 中名为 **version.bind** 的查询报告的版本。缺省值是该服务器的实际版本号。

directory 服务器的工作目录。配置文件中的任何非绝对路径名将被视为相对于该目录。大多数服务器输出文件（如 named.run）的缺省位置都是该目录。如果未指定目录，则工作目录缺省为 **(.)** ，且从中启动服务器的目录。指定的目录应该是绝对路径。

tkey-domain 追加到用 TKEY 生成的所有共享密钥的名称的域。当客户端请求 TKEY 交换时，它可能会也可能不会指定该密钥的适当名称。如果存在，共享密钥的名称将是 **client specified part + tkey-domain** 。否则，共享密钥的名称将是 **random hex digits + tkey-domain** 。大多数情况下，域名应该是服务器的域名。

tkey-dhkey **Diffie-Hellman** 密钥，服务器使用它生成与客户端共享的密钥，生成时使用 TKEY 的 **Diffie-Hellman** 模式。服务器必须能够从工作目录的文件中加载公钥和私钥。大多数情况下，密钥名应该是服务器的主机名。

dump-file 服务器使用 **rndc dumpdb** 将数据库转储到的文件的路径名。缺省值是 **named_dump.db** 。

pid-file 服务器在其中写入其进程 ID 的文件的路径名。如果未指定，则使用缺省路径名 **/var/run/named.pid** 。 **pid-file** 由需要向正在运行的名称服务器发送信号的程序使用。

statistics-file 服务器使用 **rndc** stats 在其中追加统计信息的文件的路径名。缺省值是服务器当前目录中的 **named.stats** 。

port 服务器用来接收和发送 DNS 协议通信的 UDP/TCP 端口号。缺省值是 53。该选项主要用于服务器测试；如果服务器使用 53 之外的端口，则无法与全局 DNS 进行通信。

random-device 服务器使用的信息源。转送信息缺损值主要是 DNSSEC 操作（如有符号区域的 TKEY 事务和动态更新）所需要的。该选项指定从其中读取转送信息缺损值的设备（或文件）。如果是一个文件，当该文件用尽时，需要转送信息缺损值的操作将失败。缺省值是 **/dev/random** （或等效值），如果该值不存在，则为 none。 **random-device** 选项在服务器启动时进行初始配置加载的过程中生效，并在后继的重新加载时被忽略。

布尔选项

auth-nxdomain

如果为“yes”（是），则始终在 NXDOMAIN 响应上设置 AA 位（即使服务器实际不具有权威性）。缺省值是“no”（否）。如果使用早期版本的 BIND，则可能需要将该选项设置为“yes”（是）。

dialup 如果为“yes”（是），则服务器会将所有区域当作当前正在通过命令拨号链路上的拨号执行区域传输，该拨号可以通过源自该服务器的通信来调用。这根据具体的区域类型具有不同的效果，并且会将区域维护集中，使所有工作均在较短的时间间隔内进行，即在每次心跳间隔进行一次，并且可能在每次呼叫进行一次。它也将禁止某些常规区域维护通信。缺省值是“no”（否）。 **dialup** 选项也可以在 view 和 zone 语句中指定，这种情况下，它将覆盖全局拨号选项。

如果区域是主区域，则服务器将向所有从属区域发送 NOTIFY 请求。这将触发从属区域中的区域序号检查（前提是它支持 NOTIFY），这样从属区域就可以在连接处于活动状态时验证区域。

如果区域是从属或存根区域，则服务器将禁止常规的“最新区域”（刷新）查询，除发送 NOTIFY 请求之外，仅在心跳时间间隔过期时执行这些查询。

更细微的控制可以使用 **notify** （仅发送 NOTIFY 消息）、 **notify-passive** （发送 NOTIFY 消息并禁止常规刷新查询）、 **refresh** （禁止常规刷新处理并在心跳时间间隔过期时发送刷新查询）和 **passive** （仅禁用常规刷新处理）来实现。

minimal-responses

如果为“yes”（是），服务器仅在生成响应和附加数据部分（如果它们是必需的，如委托、负响应）时将记录添加到权威机构。这可以提高服务器的性能。缺省值是“no”（否）。

notify 如果为“yes”（缺省值），将在更改服务器对其具有权威的区域时发送 DNS NOTIFY 消息。消息将发送到区域的 NS 记录中列出的服务器（SOA MNAME 字段中标识的主服务器除外）以及 **also-notify** 选项中列出的所有服务器。如果显式指定，通知消息将仅发送到使用 **also-notify** 显式列出 姆?衿鳌Ｈ绻???o”（否），则不发送任何通知消息。

notify 选项也可在 zone 语句中指定，这种情况下它将覆盖 options 语句中指定的 **notify** 。它只需要在从属区域崩溃时关闭。

recursion 如果为“yes”（是）并且 DNS 查询请求递归，则服务器将尝试回应查询。如果递归已关闭并且服务器不知道答案，它将返回转交响应。缺省值是“yes”（是）。请注意，将递归设置为“no”（否）不会阻止客户端从服务器的缓存获取数据；它仅防止因为客户端查询的影响而缓存新数据。由于服务器内部操作（如 NOTIFY 地址查找）的影响，仍可能会进行缓存。

zone-statistics

如果为“yes”（是），服务器将在缺省情况下收集服务器中所有区域上的统计数据。这些统计信息可以使用 **rndc stats** 来访问，后者会将其转储到统计信息文件中列出的文件。

provide-ixfr 子句确定在充当从属服务器的给定远程服务器请求时，充当主服务器的本地服务器是否将以递增的区域传输做出响应。如果设置为“yes”（是），则将尽可能提供递增传输。如果设置为“no”

（否），则向远程服务器的所有传输都将是非递增的。如果未设置，视图中 **provide-ixfr** 选项的值或全局选项块将用作缺省值。

request-ixfr **request-ixfr** 子句确定充当从属服务器的本地服务器是否将从充当主服务器的给定远程服务器请求递增区域传输。如果未设置，视图中 **request-ixfr** 选项的值或全局选项块将用作缺省值。

additional-from-auth，**additional-from-cache**

这些选项控制权威服务器在回答包含附加数据的查询或跟踪 CNAME 和 DNAME 链时的行为。

当这两个选项均设置为“yes”（缺省值）并且从权威数据（配置到服务器中的区域）回答查询时，将使用其他权威区域和缓存中的数据填充回复的附加数据部分。某些情况下不应该这样，例如当担心缓存的正确性时，或者在其中的从属区域可能由非可信的第三方添加或修改的服务器中。此外，通过避免搜索该附加数据，可以加快服务器操作的速度，可能的代价是附加了用于解析其他情况下将在附加部分中提供的内容的查询。例如，如果查询请求主机 foo.example.com 的 MX 记录，而找到的记录是“MX 10 mail.example.net”，通常还将在已知情况下提供 mail.example.net 的地址记录（A、 A6 和 AAAA）。将这些选项设置为“no”（否）可禁用该行为。这些选项应该在仅权威服务器或者仅权威视图中使用。如果将其设置为“no”（否）的同时没有指定递归“no”（否），则将导致服务器忽略这些选项，并记录警告消息。

指定 **additional-from-cache no** 实际上不仅会禁止将缓存用于附加数据查找，而且将禁止在查找答案时使用缓存。在缓存数据的正确性存在问题的仅权威服务器中，这通常是预期的行为。

当以非递归方式查询名称服务器来获取不位于任何服务区域顶点之下的名称时，它通常会通过“向上转交”到根服务器或查询名的某些其他已知父级的服务器来做出回应。

由于向上转交中的数据来自缓存，在指定了 **additional-from-cache no** 时，服务器将无法提供向上转交。它将以 REFUSED 响应这些查询。由于向上转交对于解析过程并不是必需的，因此这不应该造成任何问题。

match-mapped-addresses

如果为“yes”（是），IPv4 映射的 IPv6 地址将匹配与相应的 IPv4 地址相匹配的列表条目。

edns （扩展 DNS）子句确定本地服务器在与任何远程服务器通信时是否将尝试使用 EDNS。缺省值是“yes”（是）。如果为“no”（否），则在与任何远程服务器通信时将不使用 EDNS。这样，要启用服务器的 EDNS 功能，用户需要在 **server** 语句中为该服务器设置 **edns yes** 。

转发

转发工具可用于在几个服务器上创建大型的站点范围缓存，以减少通过链接到达外部名称服务器的通信量。它也可用于允许无法直接访问 Internet 但仍希望查找外部名称的服务器执行查询。转发仅在那些服务器对其没有权威并且在其缓存中没有答案的查询上进行。

forward 只有在转发程序列表不为空时，该选项才有用。缺省值 **first** 可使服务器首先查询转发程序，如果不能回答问题，服务器将查找答案本身。如果指定了 **only** ，服务器将只查询转发程序。

forwarders 指定要用于转发的 IP 地址。缺省值是空列表（无转发）。

转发只能逐个域地进行配置，从而可以通过多种方法覆盖全局转发选项。可以设置特定域使用不同的转发程序，或者使用不同的 **forward only/first:** 行为，或者根本不转发。

访问控制

服务器访问可以根据请求系统的 IP 地址来进行限制。

allow-notify 指定允许哪些主机向从属区域通知（除通知主区域之外）区域更改。 **allow-notify** 也可以在 zone 语句中指定，这种情况下它将覆盖选项 **allow-notify** 语句。它仅对于从属区域才有意义。如果未指定，缺省值是仅处理主区域中的通知消息。

allow-query 指定允许哪些主机询问普通问题。 **allow-query** 也可以在 zone 语句中指定，这种情况下它将覆盖选项 **allow-query** 语句。如果未指定，缺省值是允许所有主机中的查询。

allow-recursion 指定允许哪些主机通过该服务器执行递归查询。如果未指定，缺省值是允许所有主机中的递归查询。请注意，禁用主机的递归查询不会防止主机检索已经在服务器缓存中的数据。

allow-v6-synthesis

指定需要接收对 IPv6 查询的合成响应的主机。有关详细信息，请参阅下文。

allow-transfer 指定允许哪些主机接收服务器中的区域传输。 **allow-transfer** 也可以在 zone 语句中指定，这种情况下它将覆盖选项 **allow-transfer** 语句。如果未指定，缺省值是允许所有主机中的传输。

blackhole 指定服务器将不接受其查询或用来解析查询的地址的列表。将不响应来自这些地址的查询。缺省值是 **none** 。

接口

服务器将回答其查询的接口和端口可以使用 **listen-on** 选项指定。 **listen-on** 采用一个可选端口和一个 **address_match_list** 。服务器将在地址匹配列表所允许的所有地址上监听。如果未指定端口，则使用端口 53。

允许多个 **listen-on** 语句。例如，

```
listen-on { 5.6.7.8; };
listen-on port 1234 { !1.2.3.4; 1.2/16; };
```

将启用端口 53 上 IP 地址为 5.6.7.8 的名称服务器，以及网络 1.2 中计算机上非 1.2.3.4 地址的端口 1234 上的名称服务器。如果未指定 **listen-on** ，服务器将在所有接口上的端口 53 上监听。 **listen-on-v6** 选项用于指定服务器将在哪些端口上监听使用 IPv6 发送的传入查询。

与 IPv4 不同，服务器不会将单独的套接字绑定到每个 IPv6 接口地址。它始终会在 IPv6 通配符地址上监听。因此， **listen-on-v6** 语句的 **address_match_list** 参数只能使用以下值：

```
{ any; }
和
{ none;}
```

多个 **listen-on-v6** 选项可用于在多个端口上监听：

```
listen-on-v6 port 53 { any; };
```

listen-on-v6 port 1234 { any; };

要使服务器不在任何 IPv6 地址上监听，请使用

listen-on-v6 { none; };

如果未指定 **listen-on-v6** 语句，服务器将不在任何 IPv6 地址上监听。

查询地址

如果服务器无法回答问题，它将查询其他名称服务器。 **query-source** 将指定用于这些查询的地址和端口。对于通过 IPv6 发送的查询，存在单独的 **query-source-v6** 选项。如果地址是 * 或者被忽略，则将使用通配符 IP 地址 (**INADDR_ANY**) 。如果端口是 * 或者被忽略，则将使用随机的无特权端口。缺省地址和端口为:

query-source address * port *;
query-source-v6 address * port *

注释： **query-source** 选项中指定的地址用于 UDP 和 TCP 查询，但是端口仅应用于 UDP 查询。TCP 查询始终会使用随机的无特权端口。

区域传输

BIND 具有方便区域传输的机制，并限制传输在系统上施加的负载量。下列选项仅适用于区域传输。

also-notify 定义指定名称服务器的 IP 地址的全局列表，除了区域 NS 记录中列出的服务器之外，每当加载区域的全新副本时，还将给这些名称服务器发送 NOTIFY 消息。这有助于确保区域的副本在秘密服务器上快速汇聚。如果 **also-notify** 列表在 zone 语句中给定，它将覆盖选项 **also-notify** 语句。区域 notify 语句设置为“no”（否）时，将不会给该区域的全局 **also-notify** 列表中的 IP 地址发送 NOTIFY 消息。缺省值是空列表（无全局通知列表）。

max-transfer-time-in
运行时间长于该分钟数的入站区域传输将终止。缺省值是 120 分钟（2 小时）。

max-transfer-idle-in
在该分钟数的时间内没有进度的入站区域传输将终止。缺省值是 60 分钟（1 小时）。

max-transfer-time-out
运行时间长于该分钟数的出站区域传输将终止。缺省值是 120 分钟（2 小时）。

max-transfer-idle-out
在该分钟数的时间内没有进度的出站区域传输将终止。缺省值是 60 分钟（1 小时）。

serial-query-rate
从属服务器将定期查询主服务器，以确定区域序号是否已更改。每个这样的查询将使用从属服务器网络带宽的分钟数。为了限制所用的带宽量，BIND 9.2 将速率限制在查询发送速率。 **serial-query-rate** 选项的整数值是每秒发送的最大查询数。缺省值是 20。

transfer-format

区域传输可以使用两种不同格式发送： **one-answer** 和 **many-answers** 。传输格式选项用于在主服务器上确定所发送的格式。 **one-answer** 为传输的每个源记录使用一个 DNS 消息。 **many-answers** 会将尽量多的资源记录压缩到一条消息中。 **many-answers** 更为高效，但仅受较新的从属服务器（如 BIND 9.2、BIND 8.x 以及

BIND 4.9.x 的修补版本）支持。缺省值是 **many-answers** 。 **transfer-format** 可以逐个服务器地使用 server 语句覆盖。

transfers-in

并发运行的入站区域传输的最大数量。缺省值是 10。通过增加 **transfers-in** ，可以加速从属区域的汇聚，但是也会增加本地系统上的负载。

transfers-out

并发运行的出站区域传输的最大数量。将拒绝超过该限制的区域传输请求。缺省值是 10。

transfers-per-ns

从给定远程名称服务器中并发运行的入站区域传输的最大数量。缺省值是 2。通过增加 **transfers-per-ns** ，可以加速从属区域的汇聚，但也会增加远程名称服务器上的负载。 **transfers-per-ns** 可以使用 **server** 语句的 **transfers** 短语逐个服务器地进行覆盖。

transfer-source

transfer-source 确定哪些本地地址将绑定到用来获取由服务器传输入站的区域的 IPv4 TCP 连接。它还确定用于刷新查询和转发动态更新的源 IPv4 地址和可选的 UDP 端口。如果未设置，它将缺省为系统控制的值，该值通常是“最接近”远程端的接口地址。如果已指定，对于所传输的区域，该地址必须出现在远程端的 **allow-transfer** 选项中。该语句设置所有区域的 transfer-source ，但可以通过将 **transfer-source** 语句包括在配置文件的区域或视图块来逐个区域或逐个视图地进行覆盖。

transfer-source-v6

除了使用 IPv6 执行区域传输之外，与 transfer-source 相同。

notify-source

notify-source 确定将使用哪些本地源地址和（可选）UDP 端口来发送 NOTIFY 消息。该地址必须出现在从属服务器的主 **zone** 子句或者出现在 **allow-notify** 子句中。该语句设置所有区域的 **notify-source** ，但可以通过将 **notify-source** 语句包括在配置文件的视图或区域块来逐个视图或逐个区域地进行覆盖。

notify-source-v6

除了适用于发送到 IPv6 地址的通知消息外，与 **notify-source** 相同。

操作系统资源限制

可以限制服务器对许多系统资源的使用。指定资源限制时可以使用缩放值。例如，可以使用 1G 代替 1073741824 来指定一千兆字节的限制。“unlimited_size_spec”请求无限使用量或者可用的最大使用量。default 使用在启动服务器时强加的限制。

下列选项设置名称服务器进程的操作系统资源限制。如果使用不支持的限制，则将发出警告。

coresize 核心转储的最大大小。缺省值为 **default** 。

datasize 服务器可使用的最大数据内存量。缺省值为 **default** 。这是对服务器内存使用量的硬性限制。如果服务器尝试分配超过该限制的内存，分配将失败，进而可能使服务器无法执行 DNS 服务。因此，很少使用该选项来限制服务器使用的内存量，但它可用于提高在缺省情况下太小的操作系统数据大小限制。如果要限制服务器使用的内存量，请使用 **max-cache-size** 和 **recursive-clients** 选项。

files 服务器可以同时打开的最大文件数。缺省值是 **unlimited** 。

stacksize 服务器可使用的最大堆栈内存量。缺省值为 **default** 。

服务器资源限制

下列选项设置对服务器资源使用的限制，这些限制最初由服务器（而不是操作系统）在内部强加。

recursive-clients
服务器将代表客户端执行的最大并发递归查找数。缺省值是 1000。由于每个递归客户端按照 20 KB 的顺序使用一定量的内存，可能需要在内存有限的主机上减少 **recursive-clients** 选项的值。

tcp-clients 服务器将接受的最大并发客户端 TCP 连接数。缺省值为 100。

max-cache-size
用于服务器缓存的最大内存量（字节）。当缓存中的数据量达到该限制时，服务器将使记录提前过期，以便不超过该限制。在包含多个视图的服务器中，该限制分别应用于每个视图的缓存。缺省值是 **unlimited** ，表示仅当其 TTL 过期时才将记录从缓存中清除。

周期任务时间间隔

cleaning-interval
每隔 **cleaning-interval** 分钟，服务器将从缓存中删除过期的资源记录。缺省值是 60 分钟。如果设置为 0，则将不进行周期性清理。

heartbeat-interval
每当该时间间隔过期时，服务器将为标记为拨号的所有区域执行区域维护任务。缺省值为 60 分钟。合理的值是最多 1 天（1440 分钟）。如果设置为 0，则不为这些区域进行区域维护。

interface-interval
服务器每隔 **interface-interval** 分钟就会扫描一次网络接口。缺省值为 60 分钟。如果设置为 0，仅在加载配置文件时执行接口扫描。扫描后，监听程序将在任何新接口上启动（前提是 **listen-on** 配置允许）。将清理接口上已退出的监听程序。

sortlist 语句

对 DNS 查询的响应可能由多个资源记录 (RR) 组成，这些记录形成一个资源记录集 (RRset)。名称服务器通常将按照不确定的顺序返回 RRset 中的 RR。客户端解析程序代码应该对 RR 进行适当的重新排列，即首选使用本地网络上的任意地址。但是，并非所有解析程序都可以执行该操作或者已正确配置。当客户端使用本地服务器时，可以根据客户端的地址在服务器中执行排序。这只要求配置名称服务器，而不是配置所有客户端。 sortlist 语句（请参考下面的 sortlist 一节）采用 **address_match_list** 并对其进行解释。sortlist 中的每个顶级语句本身必须是包

含一个或两个元素的显式 **address_match_list** 。在找到匹配项之前，将对照查询的源地址检查每个顶级列表的第一个元素（可以是 IP 地址、IP 前缀、ACL 名称或嵌套的 **address_match_list** ）。

匹配查询的源地址后，如果顶级语句仅包含一个元素，则将使用匹配源地址的实际原始元素来选择地址，以响应向响应开头的移动。如果该语句是包含两个元素的列表，则以特殊的方式解释第二个元素。将给每个顶级元素分配一个距离，并将具有最短距离的响应中的地址移动到响应的开头。在下例中，从主机本身的任何地址收到的任何查询将得到首选任何本地连接网络地址的响应。接着是 192.168.1/24 网络上的地址，它之后是 192.168.2/24 或 192.168.3/24 网络，这两个网络的首选级相同。相对于 192.168.2/24 和 192.168.3/24 网络，从 192.168.1/24 网络上的主机收到的查询将首选该网络上的其他地址。从
192.168.4/24 或 192.168.5/24 网络上的主机收到的查询将仅首选其直接连接网络上的其他地址。

```
sortlist {
  { localhost;
                    // IF   the local host
    { localnets;
                    // THEN first fit on the
      192.168.1/24;
                    //   following nets
      { 192.168.2/24; 192.168.3/24; }; }; };
  { 192.168.1/24;
                    // IF   on class C 192.168.1
    { 192.168.1/24;
                    // THEN use .1, or .2 or .3
      { 192.168.2/24; 192.168.3/24; }; }; };
  { 192.168.2/24;
                    // IF   on class C 192.168.2
    { 192.168.2/24;
                    // THEN use .2, or .1 or .3
      { 192.168.1/24; 192.168.3/24; }; }; };
  { 192.168.3/24;
                    // IF   on class C 192.168.3
    { 192.168.3/24;
                    // THEN use .3, or .1 or .2
      { 192.168.1/24; 192.168.2/24; }; }; };
  { { 192.168.4/24; 192.168.5/24; };
                    // if .4 or .5, prefer that net
  };
};
```

下例将提供本地主机以及直接连接网络上的主机的合理行为。它类似于
BIND 4.9.x 中的地址排序行为。从本地主机发送到查询的响应将首选任何直接连接的网络。从直接连接网络上的

其他任何主机发送到查询的响应将首选该相同网络上的地址。对其他查询的响应将不排序。

```
sortlist {
        { localhost; localnets; };
        { localnets; };
};
```

合成 **IPv6** 响应

许多现有存根解析程序支持 RFC1886 中定义的 IPv6 DNS 查找，即使用 AAAA 记录进行向前查找，使用 ip6.int 域中的 **nibble** 标签进行向后查找，但是它们不支持 RFC2874 式查找（使用 A6 记录和 ip6.arpa 域中的二进制标签）。

对于希望继续使用这些存根解析程序而不愿切换到 BIND 9.2 轻量级解析程序的用户，BIND 9.2 提供了一种方法来自动将 RFC1886 式查找转换为 RFC2874 式查找，并以“合成”AAAA 和 PTR 记录的形式返回结果。该功能在缺省情况下禁用，可以通过将 **allow-v6-synthesis { address_match_list };** 子句添加到选项或 **view** 语句来逐个客户端地启用。启用后，递归 AAAA 查询将使服务器首先尝试 A6 查找，在其失败时再尝试 AAAA 查找。无论哪一种查找成功，均以合成 AAAA 记录集的形式返回结果。同样，ip6.int 中的递归 PTR 查询将导致使用二进制标签在 ip6.arpa 中进行查找，如果失败，则在 ip6.int 中进行另一次查找。结果将以 ip6.int 中合成 PTR 记录的形式返回。

合成记录的 TTL 为零。当前不支持合成响应的 DNSSEC 验证；因此，包含合成 RR 的响应将未设置 AD 标志。

微调

lame-ttl 设置缓存无效服务器指示的秒钟数。0 禁用缓存（不建议）。缺省值为 600（10 分钟）。最大值是 1800（30 分钟）。

max-ncache-ttl 为了减少网络通信量并提高性能，服务器将存储负答案。 **max-ncache-ttl** 用于设置这些答案在服务器中的最长保留时间（秒钟）。缺省的 **max-ncache-ttl** 是 10800 秒钟（3 小时）。 **max-ncache-ttl** 不能超过 7 天，如果设置为更大的值，将截断到 7 天。

max-cache-ttl 设置服务器缓存普通（正）答案的最长时间。缺省值是一周（7 天）。

sig-validity-interval

指定因动态更新自动生成的 DNSSEC 签名会在将来的多少天内到期。缺省值为 30 天。签名最初时间无条件地设置为当前时间之前的一小时，以允许有限的时钟偏离量。

min-refresh-time、 **max-refresh-time**,

min-retry-time、**max-retry-time**

这些选项控制服务器在刷新区域（SOA 更改查询）或重试失败传输时的行为。通常将使用区域的 SOA 值，但是这些值由主服务器设置，因此从属服务器管理员对其内容具有极少的控制。

通过这些选项，管理员可以逐个区域、逐个视图或逐个服务器地设置最短和最长的刷新和重试时间。这些选项对于主区域、从属区域和存根区域有效，它们将 SOA 刷新和重试时间限定到指定值。

统计信息文件

BIND 9.2 生成的统计信息文件与 BIND 8 生成的统计信息文件类似，但不完全相同。统计信息转储从行 +++ Statistics Dump +++ (973798949) 开始，括号内的数字是标准的 Unix 式时间戳，按照 1970 年 1 月 1 日后的秒钟数进行度量。该行之后是一系列包含计数器类型、计数器值、（可选）区域名称和（可选）视图名称的行。未列出视图和区域的行是整个服务器的全局统计信息。包含区域和视图名称的行是给定视图和区域的统计信息（对于缺省视图将省略视图名称）。统计信息转储以行 --- Statistics Dump --- (973798949) 结尾，其中的数字与开头行中的数字相同。将维护以下统计信息计数器:

success 对服务器或区域执行的成功查询数。成功的查询定义为返回 NOERROR 响应（而不是转交响应）的查询。

referral 导致转交响应的查询数。

nxrrset 导致无数据的 NOERROR 响应的查询数。

nxdomain 导致 NXDOMAIN 响应的查询数。

recursion 导致服务器执行递归来查找最终答案的查询数。

failure 导致上述响应之外的其他失败响应的查询数。

server 语句语法

```
server ip_addr {
   [ bogus yes_or_no ; ]
   [ provide-ixfr yes_or_no ; ]
   [ request-ixfr yes_or_no ; ]
   [ edns yes_or_no ; ]
   [ transfers number ; ]
   [ transfer-format ( one-answer | many-answers ) ; ]]
   [ keys { string ; [ string ; [...]] } ; ]
};
```

server 语句定义和用法

server 语句定义与远程名称服务器关联的特性。 **server** 语句可以在配置文件的顶级或者在 **view** 语句内出现。如果 **view** 语句包含一个或多个 **server** 语句，只有这些语句才应用于视图，而将忽略任何顶级语句。如果 **view** 不包含 **server** 语句，则任何顶级 **server** 语句都将用作缺省值。

如果发现远程服务器正在发送错误数据，可通过将其标记为“bogus”（伪服务器）将防止对其进行进一步的查询。bogus 的缺省值是“no”（否）。 **provide-ixfr** 子句确定在充当从属服务器的给定远程服务器请求时，充当主服务器的本地服务器是否将以递增的区域传输做出响应。如果设置为“yes”（是），则将尽可能提供递增传输。如果设置为“no”（否），则向远程服务器的所有传输都将是非递增的。如果未设置，视图中 **provide-ixfr** 选项的值或全局选项块将用作缺省值。

request-ixfr 子句确定充当从属服务器的本地服务器是否将从充当主服务器的给定远程服务器请求递增区域传输。如果未设置，视图中 **request-ixfr** 选项的值或全局选项块将用作缺省值。

对不支持 IXFR 的服务器的 **IXFR** 请求将自动回退到 AXFR。因此，无须手动列出支持 IXFR 的服务器和不支持 IXFR 的服务器；“yes”（是）全局缺省值始终有效。 **provide-ixfr** 和 **request-ixfr** 子句的目的是便于禁止使用 IXFR，即使主服务器和从属服务器均声称支持 IXFR，例如，其中一个服务器有缺陷，它在使用 IXFR 时导致数据崩溃或损坏。

edns （扩展 DNS）子句确定本地服务器在与远程服务器通信时是否将尝试使用 EDNS。缺省值是“yes”（是）。

服务器支持两种区域传输方法。第一种, **one-answer** 为传输的每个资源记录使用一条 DNS 消息。 **many-answers** 将尽可能多的资源记录压缩到一条消息中。 **many-answers** 更为高效，但已知只有 BIND 9、BIND 8.x 以及 BIND 4.9.5 修补版本才能识别。可以用 **transfer-format** 选项指定用于特定服务器的方法。如果未指定 **transfer-format**，则将使用 **options** 语句指定的 **transfer-format** 。 **transfers** 子句用于限制指定服务器中的并发入站区域数。如果未指定 **transfers** 子句，则根据 **transfers-per-ns** 选项设置限制。

keys 子句用于标识 key 语句定义的 **key_id** ，以便在与远程服务器通信时用于事务安全性。 key 语句必须位于引用它的 server 语句之前。当请求发送到远程服务器时，将使用此处指定的密钥生成请求签名，并将其追加到消息。源自远程服务器的请求不必用该密钥签名。虽然 keys 子句的语法允许使用多个密钥，但目前每个服务器仅支持一个密钥。

trusted-keys 语句语法

```
trusted-keys {
   string number number number string ;
   [ string number number number string ; [...]]
};
```

trusted-keys 语句定义和用法

trusted-keys 语句定义 DNSSEC 安全根。安全根在非权威区域的公钥已知时定义，但是无法通过 DNS 安全地获取，这可能是因为它是 DNS 根区域或者其父区域未签名。将密钥配置为可信密钥后，会将其当作已经过验证并证明是安全的。解析程序将尝试对安全根子域中的所有 DNS 数据进行 DNSSEC 验证。

trusted-keys 语句可以包含多个密钥条目，每个条目由密钥的域名、标志、协议、算法和密钥数据的基数 64 表示形式组成。

view 语句语法

```
view
 view_name [class] {
    match-clients { address_match_list } ;
    match-destinations { address_match_list } ;
    match-recursive-only { yes_or_no } ;
    [ view_option; ...]
    [ zone-statistics yes_or_no ; ]
    [ zone_statement; ...]
};
```

view 语句定义和用法

通过 view 语句，名称服务器可以根据谁提出的问题来以不同的方式回答 DNS 查询。对于实现拆分 DNS 设置而不运行多个服务器，它特别有用。每个 view 语句定义一个客户端子集将看到的 DNS 命名空间的视图。如果客户端的源 IP 地址匹配视图 **match-clients** 子句的 **address_match_list** ，其目标地址 IP 地址匹配视图 **match-destinations** 子句的 **address_match_list** ，则该客户端匹配视图。

如果未指定，则 **match-clients** 和 **match-destinations** 均缺省为匹配所有地址。视图也可以指定为 **match-recursive-only** ，这意味着只有匹配客户端中的递归请求才将匹配该视图。 view 语句的顺序很重要 - 客户端请求将在它匹配的第一个视图的环境中解析。

view 语句中定义的区域仅供匹配视图的客户端访问。通过在多个视图中定义同名的区域，可以给不同的客户端（例如拆分 DNS 设置中的“内部”和“外部”客户端）提供不同的区域数据。

options 语句中提供的多个选项也可以在 view 语句中使用，然后只在用该视图解析查询时应用。当未给定视图专用值时，options 语句中的值将用作缺省值。此外，zone 选项可以具有在 view 语句中指定的缺省值；这些视图专用缺省值优先于
options 语句中的缺省值。

视图是特定于类的。如果未给定任何类，则假定 **IN** 。请注意，由于只有 IN 类才包含编译缺省值提示，所有非 IN 视图必须包含提示区域。如果配置文件中没有 view 语句，则将自动在类 IN 中创建匹配任何客户端的缺省视图，在配置文件顶级指定的任何 zone 语句将被视作该缺省视图的一部分。如果存在任何显式的 view 语句，所有 zone 语句必须在 view 语句内出现。

下面是使用 view 语句实现的典型拆分 DNS 设置示例。

```
view "internal" {
          // This should match our internal networks.
    match-clients { 10.0.0.0/8; };
          // Provide recursive service to internal clients only.
    recursion yes;
          // Provide a complete view of the example.com zone
          // including addresses of internal hosts.
    zone "example.com" {
        type master;
        file "example-internal.db";
    };
};

view "external" {

    match-clients { any; };
          // Refuse recursive service to external clients.
    recursion no;
```

```
            // Provide a restricted view of the example.com zone
            // containing only publicly accessible hosts.
        zone "example.com" {
            type master;
            file "example-external.db";
        };
    };
```

zone 语句语法

```
zone zone_name [class] [{
    type ( master | slave | hint | stub | forward ) ;
    [ allow-notify { address_match_list } ; ]
    [ allow-query { address_match_list } ; ]
    [ allow-transfer { address_match_list } ; ]
    [ allow-update { address_match_list } ; ]
    [ update-policy { update_policy_rule [...] } ; ]
    [ allow-update-forwarding { address_match_list } ; ]
    [ also-notify { ip_addr [port ip_port] ;
                    [ ip_addr [port ip_port] ; ... ] }; ]
    [ dialup dialup_option ; ]
    [ file string ; ]
    [ forward (only|first) ; ]
    [ forwarders { ip_addr [port ip_port] ;
                  [ ip_addr [port ip_port] ; ... ] }; ]
    [ masters [port ip_port] { ip_addr [port ip_port]
                    [key key]; [...] } ; ]
    [ max-transfer-idle-in number ; ]
    [ max-transfer-idle-out number ; ]
    [ max-transfer-time-in number ; ]
    [ max-transfer-time-out number ; ]
    [ notify yes_or_no | explicit ; ]
    [ transfer-source (ip4_addr | *) [port ip_port] ; ]
    [ transfer-source-v6 (ip6_addr | *) [port ip_port] ; ]
    [ notify-source (ip4_addr | *) [port ip_port] ; ]
    [ notify-source-v6 (ip6_addr | *) [port ip_port] ; ]
    [ zone-statistics yes_or_no ; ]
    [ sig-validity-interval number ; ]
    [ database string ; ]
    [ min-refresh-time number ; ]
```

[max-refresh-time number ;]
[min-retry-time number ;]
[max-retry-time number ;]

}];

zone 语句定义和用法

Zone Types

master 服务器具有区域数据的主副本，并且能够为其提供权威性答案。

slave 从属区域是主区域的副本。主服务器列表指定从属服务器与其联系以更新其区域副本的主服务器的一个或多个 IP 地址。缺省情况下，传输从服务器上的端口 53 进行；通过在 IP 地址列表前指定端口号可为所有服务器更改该设置，也可以在 IP 地址后逐个服务器地更改该设置。主服务器的身份验证也可以使用各个服务器的 TSIG 密钥来完成。如果指定了文件，每次更改区域时，副本将写入该文件，并且将在服务器重新启动时从该文件重新加载。建议使用文件，这是因为它通常会加速服务器启动，并避免不必要地浪费带宽。如果数据库文件非常大，则建议将其放在不同的目录中。

stub 除了仅复制主区域的 NS 记录（而不是整个区域）之外，存根区域与从属区域类似。存根区域不是 DNS 的标准部分；它们是针对于 BIND 实现的功能。通过存根区域，无须在父区域中粘贴 NS 记录，但是需要在 named.conf 中维护一个存根区域条目和一组名称服务器地址。对于新配置不建议该用法，BIND 9.2 仅提供有限的支持。在 BIND 4/8 中，父区域中的区域传输包括该区域存根子级中的 NS 记录。这意味着，在某些情况下，用户可以仅在主服务器中为父区域配置子存根区域。 BIND 9 决不会这样将来自不同区域的区域数据混在一起。因此，如果服务于父区域的 BIND 9 主服务器配置了子存根区域，该父区域的所有从属服务器也需要配置相同的子存根区域。存根区域也可用于强制给定域的解析使用特定的权威服务器集。例如，专用网络上使用 RFC2157 寻址的缓存名称服务器可以用存根区域配置，让 10.in-addr.arpa 使用一组内部名称服务器作为该域的权威服务器。

forward 转发区域可用于逐个域地配置转发。 forward 类型的 zone 语句包含一个 forward 和（或） forwarders 语句，它们将应用于通过区域名指定的域中的查询。如果不存在 forwarders 语句，或者为
forwarders 提供了空列表，则将不为该域执行转发，从而抵消 options 语句中任何 forwarders 的作用。因此，如果要使用该类区域更改全局转发选项的行为（即先“首先转发到”再“仅转发”或者相反的顺序，但需要使用全局设置的相同服务器），则需要重新指定全局转发程序。

hint 初始的根名称服务器集使用提示区域来指定。服务器启动时，它将使用根提示来查找根名称服务器，并获取最新的根名称服务器列表。如果没有为类 IN 指定提示区域，服务器将使用编译时缺省的根服务器提示集。IN 之外的类没有内置缺省提示。

类

区域名称可以选择性地后接一个类。如果未指定类，则假定类 IN（对于 Internet）。这在绝大多数情况下是正确的。“hesiod”类是为 MIT
Athena 项目的信息服务而命名的。它用于共享有关各个系统数据库的信息，如用户、组、打印机等。关键字 HS 是 hesiod 的同义字。 MIT 开发的另一项协议是 CHAOSnet，它是在 70 年代中期创建的 LAN 协议。其区域数据可以使用 CHAOS 类指定。

区域选项

allow-notify 请参阅有关 *allow-notify* 的说明。

allow-query 请参阅有关 *allow-query* 的说明。

allow-transfer 请参阅有关 *allow-transfer* 的说明。

allow-update 指定允许哪些主机为主区域提交动态 DNS 更新。缺省值是拒绝来自所有主机的更新。请注意，该选项不适用于从属区域。有关详细信息，请参阅下面的“动态更新策略”。

update-policy 指定“简单安全更新”策略。有关详细信息，请参阅下面的“动态更新策略”。

allow-update-forwarding
指定允许哪些主机将动态 DNS 更新提交到从属区域以转发给主区域。缺省值是 **{ none; }**，它表示不执行更新转发。要启用更新转发，请指定 **allow-update-forwarding { any; };** 。由于更新访问控制应该由主服务器而不是从属服务器来负责，因此指定 **{ none; }** 或 **{ any; }** 之外的值通常会起反作用。请注意，如果在从属服务器上启用更新转发功能，可能会使依赖于基于非安全 IP 地址的访问控制的主服务器容易受到攻击。

also-notify 只有该区域的 notify 处于活动状态时才有意义。将接收该区域的 DNS NOTIFY 消息的计算机集由已经为该区域列出的所有名称服务器（而不是主服务器）以及用 **also-notify** 指定的 IP 地址组成。可以使用每个 **also-notify** 地址指定一个端口，将通知消息发送到缺省端口 53 之外的端口。**also-notify** 对于存根区域没有意义。缺省值是空列表。

database 指定用于存储区域数据的数据库类型。database 关键字之后的字符串解释为空格分隔的单词列表。第一个单词标识数据库类型，后继单词作为参数传递到数据库，以特定于数据库类型的方式进行解释。缺省值是 *rbt*，即 BIND 9.2 的本机内存中红黑树数据库。该数据库不采用参数。如果已经将附加数据库驱动程序链接到服务器，则可能存在其他值。

dialup 请参阅有关 **dialup** 的说明。

forward 只有在区域包含转发程序列表时才有意义。only 值将使查找在尝试转发程序而没有得到答案时失败，而 first 则允许尝试常规查找。

forwarders 用于覆盖全局转发程序的列表。如果没有在类型是 forward 的区域中指定，则将不为该区域执行转发；将不使用全局选项。

max-transfer-time-in
请参阅上面的 *max-transfer-time-in* 说明。

max-transfer-idle-in
请参阅上面的 *max-transfer-idle-in* 说明。

max-transfer-time-out
请参阅上面的 *max-transfer-time-out* 说明。

max-transfer-idle-out
请参阅上面的 *max-transfer-idle-out* 说明。

notify 请参阅上面的 *notify* 说明。

zone-statistics 如果为“yes”（是），服务器将保留该区域的统计信息，该信息可转储到服务器选项中定义的统计信息文件。

sig-validity-interval
请参阅有关 *sig-validity-interval* 的说明。

transfer-source 请参阅有关 *transfer-source* 的说明。

transfer-source-v6
请参阅有关 *transfer-source-v6* 的说明。

notify-source 请参阅有关 *notify-source* 的说明。

notify-source-v6 请参阅有关 *notify-source-v6* 的说明。

min-refresh-time, **max-refresh-time**,

min-retry-time, **max-retry-time**
请参阅以上说明。

动态更新策略

BIND 9.2 支持通过两种方法给客户端授予执行区域动态更新的权利，它们分别通过 **allow-update** 和 **update-policy** 选项配置。

allow-update 子句的工作方式与在先前 BIND 版本中的工作方式相同。它向给定客户端授予更新区域中任何名称的任何记录的权限。

update-policy 子句是在 BIND 9.2 中新增的，通过它可以对允许的更新进行更加细微的控制。将指定一组规则，其中每项规则授予或拒绝以一个或多个身份更新一个或多个名称的权限。如果动态更新请求消息已签名（即包括 TSIG 或 SIG(0) 记录），则可以确定签名方的身份。

规则在 **update-policy** 区域选项中指定，并且仅对于主区域才有意义。当存在 **update-policy** 语句时，如果同时存在 **allow-update** 语句，则是配置错误。 **update-policy** 语句仅检查消息的签名方；源地址无关紧要。

下面是一个示例规则定义：

(grant | deny) identity nametype name [types]

每项规则授予或拒绝特权。消息成功匹配规则后，将立即授予或拒绝操作，而不检查其他规则。当签名方匹配身份字段，名称匹配名称字段并且类型在类型字段中指定时，即认为匹配规则。身份字段指定名称或通配符名称。名称类型字段包括四个值：name、subdomain、wildcard 和 self：

name 在更新的名称与名称字段中的名称相同时匹配。

subdomain 在更新的名称是名称字段（包括名称本身）中名称的子域时匹配。

wildcard 在更新的名称是名称字段中通配符名称的有效扩展时匹配。

self 在更新的名称与消息签名方相同时匹配。将忽略名称字段。

如果未指定类型，规则将匹配除 SIG、NS、 SOA 和 NXT 之外的所有类型。类型可以按名称指定，其中包括“ANY”（ANY 匹配除 NXT 之外的所有类型，NXT 决不会更新）。

区域文件

资源记录的类型以及它们的使用场合：

本节介绍资源记录 (RR) 的概念，并说明按照 RFC 1034 应该在何时使用它们。

资源记录

域名标识一个节点。每个节点包含一组资源信息，该信息可能为空。与特定名称关联的资源信息集由不同的 RR 组成。RR 在集合中的顺序并不重要，并且不需要由名称服务器、解析程序或 DNS 的其他部分来保留。但是，为了进行优化，可以将多个 RR 排序，例如，指定首先尝试特定的邻近服务器。

资源记录的组成部分是：

owner name 在其中找到 RR 的域名。

type 指定该资源记录中资源类型的编码 16 位值。类型引用抽象资源。

TTL RR 的保留时间 (TTL)。该字段是以秒钟为单位的 32 位整数，主要由解析程序在缓存 RR 时使用。TTL 描述在忽略 RR 之前应该将其缓存多长时间。

class 标识协议系列或协议实例的编码 16 位值。

RDATA 描述资源的类型（并且有时包括类相关数据）。

下面是有效的 RR 类型（列出的某些类型虽然未过时，但是试验性 (x) 或历史性 (h) 的类型，它们不再普遍使用）：

A 主机地址。

A6 IPv6 地址。

AAAA IPv6 地址的过时格式。

CNAME 标识别名的规范名称。

DNAME 授权反向地址。将指定的域名替换为要查找的其他名称。在 RFC 2672 中有所说明。

HINFO 标识主机使用的 CPU 和 OS。

KEY 存储与 DNS 名称关联的公钥。

MX 标识域的邮件交换。有关详细信息，请参阅 RFC 974。

NS 域的权威名称服务器。

NXT 在 DNSSEC 中用来以安全方式指示区域中在特定时间间隔内不存在具有所有者名称的 RR，并指示对于现有名称存在哪些 RR 类型。有关详细信息，请参阅 RFC 2535。

PTR 指向域名空间其他部分的指针。

SIG(signature")"
包含在安全 DNS 中进行身份验证的数据。有关详细信息，请参阅 RFC 2535。

SOA 标识权威区域的开头。

SRV 有关知名网络服务的信息（替换 WKS）。

以下资源记录类目前在 DNS 中是有效的:

IN Internet 系统。

RDATA 描述资源的类型相关或类相关数据:

A 对于 IN 类是 32 位 IP 地址。

A6 将域名映射到 IPv6 地址，并提供前导“前缀”位间接引用。

CNAME 域名。

DNAME 提供整个域名空间子树（而不是单个节点）的备用名称。它会使所查询名称的某种后缀替换为 DNAME 记录的 RDATA 中的名称。

MX 16 位首选级值（值越小，级别越高），后接将充当所有者域邮件交换的主机名。

NS 完全限定的域名。

PTR 完全限定的域名。

SOA 若干个字段。

所有者名称通常是隐式的，而不是形成
RR 中必不可少的部分。例如，许多名称服务器在内部形成名称空间的树或散列结构以及节点外的链式 RR。剩余的 RR 部分是固定标头（类型、类、TTL，它们对于所有 RR 都是一致的）和一个变量部分 (RDATA)（它满足所述资源的需要）。

TTL 字段的含义是对 RR 可以在缓存中保留多久的时间限制。该限制不适用于区域中的权威数据；它也会超时，

但会遵循区域的刷新策略。由管理员将 TTL 分配到数据所来自的区域。虽然较短的 TTL 可以将缓存最小化，并且零 TTL 可禁止缓存，但是考虑到 Internet 性能的现实情况，建议对于典型的主机按日期顺序对这些时间排序。如果可以预期更改，可以在更改之前减小 TTL，以尽量减少更改过程中的不一致性，然后在更改后增加到它原来的值。

RR 中 RDATA 部分的数据将作为二进制字符串和域名的组合来传送。域名经常用作指向 DNS 中其他数据的“指针”。

RR 的文本表达式

RR 以二进制形式在 DNS 协议数据包中表示，并且当存储在名称服务器或解析程序时通常以高度编码的形式表示。在 RFC 1034 提供的示例中，为了显示 RR 的内容，使用了与主文件中所用的样式类似的样式。在这种格式中，虽然可以使用括号实现连续行，但大多数 RR 仍显示在一行上。

行的开头提供 RR 的所有者。如果某行以空格开头，则假定所有者与先前 RR 的所有者相同。为了提高可读性，通常会包括空白行。

在所有者之后，列出了 RR 的 TTL、类型和类。类和类型使用上述的助记符，而 TTL 是类型字段前的整数。为了避免在分析过程中出现歧义，类型和类助记符是分开的，TTL 是整数，类型助记符始终位于最后。为了明确起见，示例中通常会省略 IN 类和 TTL 值。

RR 的资源数据或 RDATA 部分利用有关典型数据表示形式的信息来给定。

例如，消息中传送的 RR 可以显示为：

```
ISI.EDU.MX10 VENERA.ISI.EDU.
MX10 VAXA.ISI.EDU
VENERA.ISI.EDUA128.9.0.32
A10.1.0.52
VAXA.ISI.EDUA10.2.0.27
A128.9.0.33
```

MX RR 包含一个 RDATA 部分，它由一个 16 位数字后接域名组成。地址 RR 使用标准的 IP 地址格式来包含 32 位 Internet 地址。

该示例显示六个 RR，三个域名中的每一个包含两个 RR。类似的 RR 也可显示为：

```
XX.LCS.MIT.EDU. INA10.0.0.44
CHAMIT.EDU. 2420
```

该示例显示 XX.LCS.MIT.EDU 的两个地址，每个地址属于不同的类。

MX 记录

如上所述，域服务器将信息存储为一系列资源记录，每个记录包含有关给定域名（它通常是但非始终是主机）的特定信息。要了解 RR，最简单的方法是将其当作数据的类型对，即与相关数据匹配并随某些附加类型信息存储的域名，它可帮助系统确定

RR 在何时是相关的。

MX 记录用于控制电子邮件的发送。该记录中指定的数据是优先级和域名。优先级控制尝试邮件发送的顺序，数字最小最先尝试。如果两个优先级相同，则随机选择服务器。如果给定优先级的服务器没有响应，则邮件传输代理将退到下一个最大的优先级。优先级数字不具有任何绝对意义 - 它们仅相对于该域名的其他 MX 记录才是相关的。给定的域名是邮件将发送到的计算机。它必须具有关联的 A 记录 - CNAME 是不够的。

对于给定域，如果存在 CNAME 记录和 MX 记录，则 MX 记录出错，将被忽略。邮件将发送到 CNAME 所指的 MX 记录中指定的服务器。

```
example.com.INMX10mail.example.com.
INMX10mail2.example.com.
INMX20mail.backup.org.
mail.example.com.INA10.0.0.1
mail2.example.com.INA10.0.0.2
```

例如：

将尝试向 mail.example.com 和 mail2.example.com （按任意顺序）发送邮件，如果均失败，则尝试发送到 mail.backup.org。

设置 TTL

RR 字段的 TTL 是以秒钟单位表示的 32 位整数，主要由解析程序在缓存 RR 时使用。 TTL 描述在忽略 RR 之前应该将其缓存多长时间。目前区域文件中会使用以下三种 TTL。

SOA SOA 中的最后一个字段是负缓存 TTL。它控制其他服务器会将您的无此域 (NXDOMAIN) 响应缓存多长时间。最长负缓存时间是 3 小时 (3h)。

$TTL 区域文件顶部的 **$TTL** 指令（位于 SOA 之前）为未设置特定 TTL 的每个 RR 提供缺省 TTL。

RR TTLs 每个 RR 均可将 TTL 当作 RR 中的第二个字段，它控制其他服务器可以将其缓存多长时间。

虽然可以显式指定单位（如 1h30m），所有这些 TTL 均缺省为使用秒钟作为单位。

IPv4 中的反向映射

反向名称解析（即从 IP 地址转换为名称）通过 in-addr.arpa 域和 PTR 记录来实现。 in-addr.arpa 域中的条目按照从最无意义到最有意义的顺序创建，并按从左到右的顺序来读取。该顺序与通常的 IP 地址写入顺序相反。因此，具有 IP 地址 10.1.2.3 的计算机将具有对应的 in-addr.arpa 名称 3.2.1.10.in-addr.arpa。该名称应该包含 PTR 资源记录，其数据字段是计算机的名称，或者，当计算机具有多个名称时，也可以包含多个 PTR 记录。例如，在 [example.com] 中

```
domain:
    $ORIGIN2.1.10.in-addr.arpa
    3IN PTR foo.example.com.
```

注释：示例中的 **$ORIGIN** 行仅用于为示例提供环境，实际使用时它们不必出现。在这里使用它们只是为了指示示例相对于列出的来源。

其他区域文件指令

主文件格式最初在 RFC 1035 中定义，随后得到了扩展。虽然主文件格式本身是与类无关的，但是一个主文件中的所有记录必须属于同一个类。主文件指令包括 **$ORIGIN** 、 **$INCLUDE** 和 **$TTL** 。

$ORIGIN 指令

语法：

$ORIGIN *domain-name* [*comment*]

$ORIGIN 设置将追加到任何不合格记录的域名。当首次读入区域时，存在隐式的 **$ORIGIN** *zone-name* 。当前 **$ORIGIN** 将追加到在 **$ORIGIN** 参数中指定的域（如果它不是绝对的）。

```
$ORIGIN example.com.
WWW    CNAME   MAIN-SERVER
```

等效于：

```
WWW.EXAMPLE.BOM. CNAME MAIN-SERVER.EXAMPLE.BOM.
```

$INCLUDE 指令

语法：

$INCLUDE filename [**origin**] [*comment*]

读取和处理文件的文件名，将其当作它在此处包括到文件中。如果指定了来源，则以将 **$ORIGIN** 设置为该值的方式处理文件，否则会使用当前 **$ORIGIN** 。来源和当前域名将恢复到 **$INCLUDE**once**the**file**has**been**read** 之前的值。

注释：RFC 1035 规定应该在 **$INCLUDE** 之后恢复当前来源，但是对于是否还应该恢复当前域名未作规定。BIND 9 将恢复这两者。这可以解释为与 RFC 1035 的不同之处和（或）一项功能。

$TTL 指令

语法：

$TTL *default-ttl* [*comment*]

为包含未定义的 TTL 的后继记录设置缺省保留时间 (TTL)。有效 TTL 的范围是 0-2147483647 秒。 **$TTL** 在 RFC 2308 中定义。

BIND 主文件扩展：**$GENERATE** 指令

语法：

$GENERATE *range lhs type rhs* [*comment*]

$GENERATE 用于创建一系列仅通过迭代符相互区别的资源记录。 **$GENERATE** 可用于轻松地生成支持 RFC

2317：无类 IN-ADDR.ARPA 委托中所述的 sub /24 反向委托所需的记录集。

$ORIGIN 0.0.192.IN-ADDR.ARPA.
$GENERATE 1-2 0 NS SERVER$.EXAMPLE.
$GENERATE 1-127 $ CNAME $.0

等效于：

0.0.0.192.IN-ADDR.ARPA NS SERVER1.EXAMPLE.
0.0.0.192.IN-ADDR.ARPA NS SERVER2.EXAMPLE.
1.0.0.192.IN-ADDR.ARPA CNAME 1.0.0.0.192.IN-ADDR.ARPA
2.0.0.192.IN-ADDR.ARPA CNAME 2.0.0.0.192.IN-ADDR.ARPA
...
127.0.0.192.IN-ADDR.ARPA CNAME 127.0.0.0.192.IN-ADDR.ARPA
.

range 它可以是以下两种格式中的任一种：start-stop 或 start-stop/step。如果使用第一种格式，则 step 设置为 1。所有 start、stop 和 step 必须是正数。

lhslhs 描述要创建的资源记录的所有者名称。lhs 侧的任何单个 $ 符号均将替换为迭代符的值。

要在输出中获取 $，需要使用反斜杠 \ 将 $ 转义，例如，\$。$ 可以选择性地后接修饰符，它们更改迭代符偏移、字段宽度和基数。修饰符通过 { 在 $ 后直接添加 ${偏移[,宽度[,基数]]} 引入，例如，${-20,3,d} 将当前值减去 20，在宽度为 3 的零填充字段中输出十进制形式的结果。可用的输出格式为 **decimal (d)** 、 **octal (o)** 和 **hexadecimal** （x 或大写的 X）。缺省修饰符是 **${0,0,d}** 。如果 lhs 不是绝对的，则将当前 $ORIGIN 追加到该名称。

为了与早期版本兼容，仍将识别 $$，它指示输出中的文字 $。

type 目前仅支持 PTR、CNAME、DNAME、A、AAAA 和 NS。

rhsrhs 是域名。它的处理方式与 *lhs* 类似。

$GENERATE 指令是 BIND 扩展，不属于标准区域文件格式的一部分。

警告

HP-UX 11i v1.0 支持 IPv6，并安装了可选的 IPv6 软件。当前，运行 HP-UX 11i v1.6 的系统上不支持 IPv6。

作者

named.conf 由 Internet Software Consortium (ISC) 开发。

另请参阅

kill(1)、hosts_to_named(1M)、sig_named(1M)、signal(2)、gethostent(3N)、resolver(3N)、resolver(4)、hostname(5)、RFC 882、RFC 883、RFC 973、RFC 974、RFC 1032、RFC 1033、RFC 1034、RFC 1035、RFC 1123。

名称

netconfig - 网络配置数据库

概要

/etc/netconfig

说明

网络配置数据库 **/etc/netconfig** 是一个系统文件，用于存储与系统相连接的网络的有关信息。 **netconfig** 数据库以及访问这个数据库的例行程序（请参阅 *getnetconfig*(3N) ）是网络选择组件的组成部分。网络选择组件还包括 **getnetpath()** 例行程序，以提供应用程序专用的网络搜索路径。这些例行程序根据环境变量 *NETPATH* 访问 **netconfig** 数据库（请参阅 *environ*(5) ）。

netconfig 包含系统上每个可用网络的条目。条目之间用换行符分隔。字段之间用空白字符分隔，其顺序如下所述。空白字符可以是嵌入的 空格或 制表符。 **/etc/netconfig** 中第一列以 **#** （井号）开头的行被视为注释。

netconfig 数据库中的每个有效行都对应于一个可用的传输。各条目均使用以下形式：

network_ID semantics_flag protocol_family protocol_name network_device translation_libraries

network_ID 用于唯一标识网络的字符串。 *network_ID* 由非空字符组成，其长度不小于 1。最大长度未指定。这个命名空间是本地有效的，本地系统管理员是命名授权者。系统上的所有 *network_ID* 都必须是唯一的。

semantics *semantics* 字段是标识网络的“语义”的字符串，即通过对其提供的服务接口进行标识，以表示它所支持的一组服务。 *semantics* 是必需的。可识别以下语义：

tpi_clts 传输提供者接口，无连接

tpi_cots_ord 传输提供者接口，面向连接，支持有序释放。

flag *flag* 字段记录网络的二元值（“true”和“false”）属性。 *flag* 是由多个字符组成的字符串，每个字符表示相应属性的值。如果有字符，则表示相应的属性为“true”。如果没有字符，则表示属性为“false”。“**-**“”表示没有任何属性。当前只识别一个字符：

v 可见（“缺省”）网络。没有设置环境变量 *NETPATH* 时，使用这个字符。

protocol_family

protocol_family 和 *protocol_name* 字段用于协议特定的应用程序。

protocol_family 字段包含一个标识协议族的字符串。 *protocol_family* 标识符遵循与 *network_ID* 相同的规则；字符串由非空字符组成，其长度不小于 1，最大长度未指定。 *protocol_family* 字段中的 A 表示未应用任何协议族标识符（网络用作实验目的）。以下是一个协议族示例：

inet 网络互联协议：UDP、TCP 等。

protocol_name

protocol_name 字段包含一个标识协议的字符串。 *protocol_name* 标识符遵循与 *network_ID* 相同的规则，即字符串由非空字符组成，长度不小于 1，最大长度未指定。 **-** 表示未应用所列的任何名称。

可识别以下协议名。

tcp 传输控制协议

udp 用户数据报协议

network_device

network_device 是用于连接传输提供方的设备的完整路径名。一般情况下，设备位于 */dev* 目录。必须指定 *network_device* 。

translation_libraries

name-to-address translation libraries 支持网络的“目录服务”（名称-地址映射服务）。这个字段中的 **-** 表示没有任何 *translation_libraries* 。对于协议族 *inet* 的网络，它具有特殊含义：它的名称-地址映射由名称服务交换根据 **switch()** （请参阅 *nsswitch.conf*(4) ）中的 *hosts* 和 *services* 条目来提供。对于其他协议族的网络，**-** 表示非功能性的名称-地址映射。否则，这个字段包含一列用逗号分隔的、指向动态链接库的路径名。库的路径名可能是绝对路径或相对路径。

基于 Itanium(R) 的库位于 **/usr/lib/hpux**[**32**|**64**] 目录。为了向后兼容， **/usr/lib** 中提供了相应的链接。例如， **/usr/lib/libstraddr.so.1** 是指向 **/usr/lib/libstraddr.1** 的链接。如果库字段被修改，而且提供了基于 Itanium 的库 和基于 PA-RISC 的库，则必须在 **/usr/lib** 目录中创建链接，以提供对 PA-RISC 库命名约定的向后兼容。如果为特定服务指定了库的绝对路径名，使用该服务的应用只有在相应的体系结构下才能正常工作。

每个字段都对应于 **struct netconfig** 结构中的一个元素。本联机帮助页中的 **struct netconfig** 和标识符定义在 **<netconfig.h>** 之中。该结构包括下列成员：

char *nc_netid	网络 ID，包括 NULL 终止符。
unsigned long nc_semantics	语义。
unsigned long nc_flag	标志。
char *nc_protofmly	协议族。
char *nc_proto	协议名称。
char *nc_device	网络设备的完整路径。
unsigned long nc_nlookups	目录查找库的数量。
char **nc_lookups	名称-地址转换库的名称。
unsigned long nc_unused[9]	为以后的扩展保留。

根据前面所标识的语义， *nc_semantics* 字段可以接受以下值：

NC_TPI_CLTS
NC_TPI_COTS_ORD

nc_flag 字段是位字段。下面是一个可识别的位字段，它与前面标识的属性相对应。 **NC_NOFLAG** 表示没有任何

属性。

NC_VISIBLE

举例

下面是一个 **netconfig** 文件示例:

```
#
# The 'Network Configuration' File.
#
# Each entry is of the form:
#
#   <network_id> <semantics> <flags> <protofamily> <protoname> <device> \
#        <nametoaddr_libs>
#
# The '-' in <nametoaddr_libs> for inet family transports indicates
# redirection to the name service switch policies for 'hosts' and
# 'services'. The '-' may be replaced by nametoaddr libraries that
# comply with the SVr4 specs, in which case the name service switch
# will not be used for netdir_getbyname, netdir_getbyaddr,
# gethostbyname, gethostbyaddr, getservbyname, and getservbyport.
# There are no nametoaddr_libs for the inet family, and currently
# nametoaddr_libs are not supported.
#
udp  tpi_clts      v     inet    udp    /dev/udp    -
tcp  tpi_cots_ord  v     inet    tcp    /dev/tcp    -
```

作者

netconfig 由 Sun Microsystems, Inc. 开发。

文件

<netconfig.h>

/etc/netconfig

另请参阅

getnetconfig(3N)、 getnetpath(3N)、 nsswitch.conf(4)。

名称

netgroup - 网络组列表

说明

文件 **/etc/netgroup** 定义了网络范围组，用于在执行远程挂接、远程登录和远程 Shell 时，进行权限检查。对于远程挂接， **netgroup** 中的信息按计算机分组；对于远程登录和远程 Shell，则按用户分组。文件 **netgroup** 中的每一行定义了一个组，格式如下：

groupname member1 member2 ...

其中， member *i* 或是其他的组名，或是一个三元组。

(*hostname, username, domainname***)**

如果这三个字段中任一个为空，则表示一个通配符。因此，

universal (,,)

定义了一个属于所有用户的组。若字段名不是以字母、数字或下划线开头（如 **-** ），则不匹配任何值。例如，请看下列条目。

justmachines (analytica,-,YOURDOMAIN)
justpeople (-,root,YOURDOMAIN)

计算机 **analytica** 属于 **YOURDOMAIN** 域中的 **justmachines** 组，但组中不包括任何用户。同样，用户 **root** 属于 **YOURDOMAIN** 域中的 **justpeople** 组，但组中不包括任何计算机。

注意，域名字段必须与当前域名（由 **domainname** 命令返回）匹配，否则条目将不匹配。另外，远程挂接将忽略用户名字段。仅主机名和域名可用。

网络信息服务 (NIS) 可提供网络组服务。当使用时，它们将保存在下列 NIS 映射中：

netgroup
netgroup.byuser
netgroup.byhost

有关网络信息服务概述的详细信息，请参考 *ypserv*(1M) 和 *ypfiles*(4) 。

作者

netgroup 由 Sun Microsystems, Inc. 开发。

文件

/etc/netgroup

另请参阅

makedbm(1M)、 mountd(1M)、 ypmake(1M)、 ypserv(1M)、 getnetgrent(3C)、 hosts.equiv(4)、 ypfiles(4)。

《Installing and Administering NFS Services》，第 7 章：“NIS配置”。

名称

netrc: .netrc - ftp、rexec 和 rexec() 的登录信息

说明

.netrc 文件包含 **ftp** 自动登录进程、 **rexec()** 库例行程序和 **rexec** 命令（请参阅 *ftp*(1) 、 *rexec*(3N) 和 *remsh*(1) ）分别使用的登录和初始化信息。该文件是可选的。如果有该文件，那么它一定位于用户的主目录。

如果 **.netrc** 文件包含需要使用的口令或帐户信息，而不使用匿名 **ftp** ，那么文件的所有者必须与当前进程的有效用户组 ID 相匹配。它的组读取、写入和执行模式位以及其他位必须全部是零，而且其所有者必须能够读取它。否则，文件将被忽略。

文件可包含以下标记，标记之间用空白字符（空格、制表符或换行符）或逗号 (**,**) 分隔：若要包含作为标记的组成部分的逗号，应将标记用引号 (**"**) 引起来。

machine *name*
: 标识远程计算机的名称。自动登录进程在 **.netrc** 文件中搜索与 **ftp** 命令行指定的远程计算机相匹配的 **machine** 标记，并将标记作为 **ftp open** 命令参数或 **rexec()** 的 ***ahost** 参数。进行匹配后，将处理其后的 **.netrc** 标记，当到达文件末尾或遇到另一个 **machine** 标记或 **default** 标记后，处理将停止。

  如果远程计算机有一个主机名别名，而且标准名与别名同时存在于 **.netrc** 文件之中， **ftp** 客户端在搜索 **.netrc** 文件时将优先使用标准名，而不是优先使用别名。如果别名作为条目提供给 **ftp open** 命令，而且从 **.netrc** 文件的开头向底部搜索，如果 ftp 客户端在发现正式主机名之前先发现主机名别名，那么它将使用别名条目。但是，如果它首先发现标准主机名，那么它将使用标准主机名条目，即使主机名别名在 **.netrc** 文件中。因此，标准主机名的高优先级要求当别名存在时应在 **ftp** 的 **.netrc** 文件中最后存放标准主机名条目。

default
: 除了 **default** 可以匹配任何名称之外，其余与 **machine** *name* 相同。只能有一个 **default** 标记，而且必须位于所有 **machine** 标记之后。它通常用于 **ftp** ，如下所示：

  default login anonymous password user@site

  这使得可以自动匿名 **ftp** 登录到未在 **.netrc** 中指定的主机。若要禁用自动登录，**ftp** 可以使用 **-n** 标志覆盖它。

login *name*
: 标识远程计算机上的用户。如果有此标记， **ftp** 或 **rexec()** 使用指定名称初始化自动登录进程。如果此标记与 **rexec -l** 命令选项使用的用户名相匹配，或者缺省情况下，与本地用户名相匹配， **rexec** 使用 **password** 标记（如果有）。

password *string*
: 提供一个口令。如果有此标记，而且远程服务器要求自动登录进程具有口令，则自动登录进程提供指定的字符串，请注意，如果这个标记存在于 **.netrc** 文件之中而且供 **anonymous** 除外的任何用户使用，而且如果除所有者以外的任何用户都可以读取 **.netrc** 文件， **ftp** 将异常中止自动登录进程。另外请注意， **.netrc** 中的口令没有加密。

account *string* 为 **ftp** 登录提供其他帐户口令。如果有这个标记，自动登录进程会在远程服务器要求输入其他帐户口令时提供指定的字符串；或者，自动登录进程在远程服务器不作任何要求时初始化 **acct** 命令。

macdef *name* 定义一个 **ftp** 宏。这个标记与 **ftp macdef** 命令相似。宏用特定的名称来定义；其内容以下一 **.netrc** 行开始，并一直延续，直到遇到一个空行（连续的换行符）。如果定义了名为 **init** 的宏，它将作为 **ftp** 自动登录进程的最后一个步骤自动执行。

举例

以下是 **hpxdzg** 主机的一个有效条目，这个主机有一个口令为 **sesame**的 **guest** 帐户：

machine hpxdzg login guest password sesame

警告

文件中的未加密口令可能导致安全风险。

作者

netrc 由加州大学伯克利分校开发。

文件

$HOME/.netrc

另请参阅

ftp(1)、 remsh(1)、 rexec(3N)。

名称

nettlgen.conf - 网络跟踪/日志和内核日志配置文件

概要

/etc/nettlgen.conf

说明

/etc/nettlgen.conf 是通用网络跟踪/日志命令和内核日志命令等命令的配置文件，它包含 **nettl** 、 **kl** 和 **netfmt** (请参阅 *nettl*(1M) 、 *kl*(1M) 和 *netfmt*(1M) ）命令所使用的配置信息。 **nettlconf** 命令（请参阅 *nettlconf*(1M) ）可维护这个文件中的网络日志、内核日志及子系统数据，允许子系统安全地添加、修改或删除该文件中的现有条目。**nettlconf** 命令还允许子系统定制日志资源使用参数和文件名。只有 **nettlconf** 命令才能修改这个文件。

文件由记录构成，记录包含用逗号 (**:**) 分隔的字段。文件中的每行都是一条唯一的记录，它可能包含全局网络信息、内核日志信息或子系统信息。记录的第一个字段是标记字段，它标识记录内所包含的信息类型。 **LOG** 标记标识全局网络日志信息； **KL** 标记标识全局内核日志信息； **SS** 标记标识子系统信息。空行或以 **#** 开头的行被忽略。

日志记录

日志记录定义用于配置日志缺省选项（如日志文件名以及是否启动或关闭控制台日志等）的静态信息。请注意，只有文件中的最后一条日志记录会被使用，前面的日志记录被忽略。用户可以使用 **nettlconf** 命令来修改网络日志信息，以满足他们的特定需求。为了使网络日志信息的更改能够生效，管理员必须使用 **nettl** 命令停止网络跟踪/日志设备，然后重新启动。

日志记录字段如下所示：

字段 编号	名称	说明
1	tag	包含 **LOG** 标记字符串。
2	Console Logging Flag	如果需要启用控制台日志，则设置为 1，否则设置为 0。
3	Log Port Size	为内部日志消息缓冲区保留的内存容量。以千字节为单位。有效范围是 1 - 32。缺省值是 8。
4	Maximum Log File Space	确定所允许的最大日志文件空间。以千字节为单位。它的值是两个容量相等的日志文件的大小之和。有效范围是 1-10240。缺省值是 1000。

5	Log File prefix	日志文件的路径和文件名，不带类型和时间扩展（.LOG0x，x 为 0 或 1）。
6	Console Filter File	控制台日志使用的过滤器配置文件的文件名。

“**Console Logging Flag**”确定在启动网络跟踪/日志工具时，是否启用控制台日志。控制台日志使用由 “**Console Filter File**”命名的文件中指定的条件，在系统控制台上显示日志消息。如果没有控制台，或者不需要控制台日志功能，可以使用 **nettlconf** 命令关闭这个功能。在系统启动期间，将始终更新 “**Console Logging Flag**”，以反映 **/etc/rc.config.d/nettl** 文件中 *NETTL_CONSOLE* 变量的值。

如果控制台日志的特殊简明形式所提供的信息不能满足需要，可以关闭控制台日志，然后用指定了所要使用的过滤器的选项文件来启动格式化程序（请参阅 *netfmt*(1M)）。

“**Log Port Size**”定义日志队列中可能的未完成消息数。日志使用 256 字节的缓冲区。这里选择的数字表明应分配多少千字节的空间。缺省大小是 8192 字节（8 的倍数），它被分成 32 个 256 字节的字节块。系统保留第一个块，其余 31 个块用于保存日志消息。每个日志消息以一个新的块开始，占用 64 字节。此外，每个块还要使用 8 个字节的开销。可以存储的最大日志消息为 7624 字节 ((31 ∗ 256) - (31 ∗ 8) - 64)。大多数网络日志消息是相当小的，因此选择 8K 的缓冲区足够网络日志工具保存大量消息。

“**Maximum Log File Space**”确定所允许的最大日志文件空间。日志文件被分为两个部分。当单个日志文件达到这里指定的最大值的一半时，日志系统删除任何现有的旧文件，并将当前文件改名为旧文件，然后启用一个新文件。缺省规范允许存储总共 1 兆字节的网络日志文件（每个文件不超过 500 千字节）。由于日志记录并不频繁，而且网络日志消息通常很小，所以这个容量对于所有需求来说都是有余的。文件空间的占用率取决于多方面的因素，如子系统启用的网络日志级别、网络的流量、连接的频率等；它是难以预测的。

“**Console Filter File**”指定包含了格式化程序过滤器的文件的文件名，控制台日志使用格式化程序过滤器。这个文件包含一些过滤器，这些过滤器控制在控制台上显示的记录信息。文件的语法与过滤器配置文件相同， 过滤器配置文件通常是与 **netfmt** 命令一起使用的。更多详细信息请参阅介绍过滤器配置文件的 *netfmt*(1M)。

如果没有控制台过滤器文件，将使用一组缺省过滤器（这组缺省过滤器将在控制台上显示 DISASTER 消息）来创建一个特殊文件。如果控制台过滤器文件存在而且它包含一个 *time_from* 过滤器，那么每次启动 **nettl** 时都将更新 *time_of_day* 和 *day_of_year* 字段。

“**Console Filter File**”字段是可选的。如果省略，将使用 **/var/adm/conslog.opts** 。

KL 记录

内核记录的字段如下所示：

字段编号	名称	说明
1	Tag	包含 **KL** 标记字符串。

2	KL Minimum (initial) Queue Size	可以同时驻留内存的最小（初始）消息数。有效范围是 100-10000，缺省值是 1000。
3	KL Maximum Queue Size	可以同时驻留内存的最大消息数。有效范围是 100 - 10000。缺省值是 1000。
4	Maximum KL File Space	确定允许范围内的最大日志文件空间。这个值是指两个容量相等的日志文件的大小之和。有效范围是 8K - 1024M。缺省值是 1M。
5	Log File Prefix	日志文件的路径和文件名，不带类型和时间扩展 (.KLOG0x，x 为 0 或 1)。
6	Write To Disk Flag	如果需要在启动 KL 工具时启用消息写入磁盘功能，应设置为 1，否则设置为 0。

“**KL Minimum and Maximum Queue Size**”定义内核日志队列中可能的未完成消息数。缺省的大小为 1000。

“**Maximum Log File Space**”确定所允许的最大日志文件的大小。日志文件分为两个部分。当单个文件达到这里指定的最大值时，内核日志系统删除任何现有的旧文件，并将当前文件名改名为旧文件，然后启用一个新文件。缺省规范允许一个日志文件占用 1 兆字节的内核日志文件存储空间。文件空间的占用率取决于多个因素，包括为每个子系统启用的内核日志级别、系统的负载、中央处理单元 (CPU) 的数量等，它是很难预测的。我们可以使用前缀 **K** 和 **M** 来说明它的大小单位是千字节还是兆字节。例如 16K = 16384 和 4M = 4096K。

“**Write To Disk Flag**”确定是否在启动内核日志工具时启用内核日志消息写入磁盘功能。写入磁盘功能用于将存储在内核日志消息中的信息保存到非易失性存储设备上，并永久记录内核活动。

子系统记录

子系统记录定义子系统所需的信息，它具有包括标记字段在内的 10 个字段。各字段之间用逗号 (**:**) 分隔，因此字段内不能含有逗号。空字段可用字符串 **NULL** 表示。注释：子系统记录内的信息更改只能在安装系统时由子系统使用 **nettlconf** 命令来完成。除非在 HP 技术支持代表的指导下进行，用户不能自行更改信息。

子系统记录的字段如下：

字段编号	名称	说明
1	tag	包含 **SS** 标记字符串。

2	Subsystem ID	取值范围为 0-1023 的整数。网络跟踪/日志工具支持范围 0-511 中的子系统。内核日志工具支持 512-1023 的子系统。这个编号由 HP 设定，而且不能更改。
3	Subsystem Mnemonic	由字母、数字和下划线字符组成的文本字符串。这个字符串是在出厂时设定的，它是不允许更改的。
4	Initial Log Level	初始化网络跟踪/日志工具或内核日志工具时，子系统的日志级别。根据子系统是否被 NetTL 工具或 KL 工具支持，它的取值会有所不同。请参阅后面的解释。
5	Subsystem Type	如果子系统基于流，而且存在于内核中，那么应将值设置为 **s**，如果子系统不基于流，而且存在于内核中，应将值设置为 **k**，如果两者都不是，则应设置为 **u**。如果子系统的 ID 是范围 512-1023 内的值，如 KL 工具支持的子系统，应将子系统类型设置为**k**。
6	Subformatter Shared Library	包含子格式化函数的共享库文件的文件名，后面介绍了子格式化函数。
7	Subformatter Message Catalog	为子系统格式化数据时所使用的消息清单基名。
8	Subformatter Function	为子系统格式化数据时需要调用的子格式化库中的 C 语言函数。
9	Subformatter Options	为了获取子系统的过滤器选项而需要调用的子格式化库中的 C 语言函数。
10	Group Name	一个文本字符串，它用于格式化输出中的标头行。

缺省日志级别的推荐设置由产品的配置脚本设定。可用的级别包括 Disaster(8)、Error(4)、Warning(2) 和 Informative(1)。可以通过添加编号来组合级别。 Disaster 和 Error 一起使用就使编号成为 12。如果需要在初始化时设置其他的日志级别，用户可以自行更改。 注释：对于 KL 子系统，如果所有日志级别比 **SS** 行中的日志级别

值严重性更高，则将启用日志。

对于 NetTL 工具所支持的子系统，使用 **nettl -log** 命令可以在运行时改变日志级别；对 KL 工具所支持的子系统，使用 **kl -l** 命令可以在运行时改变日志级别。

如果子格式化库文件名不包含绝对路径，将假设文件位于 **/usr/lib** 目录。子格式化库 必须是一个共享库。

外部语言环境影响

消息清单位于 **NLSPATH** 环境变量所确定的路径。缺省的消息清单位于 **/usr/lib/nls/%L/%N.cat** 中，此处用 **LANG** 环境变量替换 **%L** 字段，并用该环境变量中指定的名称代替 **%N** 字段。

举例

下面的例子显示缺省网络日志信息。它启用了控制台日志，使用 8 千字节以保存日志消息；日志文件总空间限定为 1000 千字节（每个文件 500 千字节）；日志文件是 **/var/adm/nettl.LOG000** 和 **/var/adm/nettl.LOG001** ；控制台日志过滤器文件是 **/var/adm/conslog.opts** 。最新的数据总是存放在 **.LOG000** 文件之中。

```
#
# LOG INFORMATION
#

LOG:1:8:1000:/var/adm/nettl:/var/adm/conslog.opts
```

下面的例子关闭了控制台日志，并将日志文件空间的大小限定为 100 千字节。 其他值与缺省值相同。

```
#
# LOG INFORMATION
#

LOG:0:8:100:/var/adm/nettl:/var/adm/conslog.opts
```

下面的例子介绍了缺省的内核日志信息。内核队列保可以保留 1000 条消息；日志文件的总空间不超过 1M 字节（每个文件 512 千字节）；内核日志文件是 **/var/adm/kl.KLOG0** 和 **/var/adm/kl.KLOG1** ； *write to disk flag* 设置为 0。最新的数据存储在 **.KLOG0** 文件中。

```
#
# KL INFORMATION
#

KL:1000:1000:1M:/var/adm/kl:0
```

下面的例子在 KL 工具启动时打开 *write to disk* 选项，将内核日志队列的大小减小为 500 条消息，并将日志文件空间减小为 5M 字节。其他值与缺省值相同。

```
#
# KL INFORMATION
#

KL:500:5000:5M:/var/adm/kl:1
```

下面的例子介绍一个典型的子系统记录。用户不能更改这些记录，但是在安装产品时，子系统可以使用 **nettlconf** 来进行设置。

```
#
# TEST NetTL SUBSYSTEMS
#

SS:96:TEST_ID_1:8:u:NULL:netfmt:subsys_GENERIC_FORMAT: \
ss_96_go:FORMATTER
SS:97:TEST_ID_2:8:u:NULL:netfmt:subsys_GENERIC_FORMAT: \
ss_97_go:FORMATTER

#
# Test KL Subsystems
#

SS:538:Test_ID_8:8:k:libklfmt.sl:klfmt:subsys_kl_format: \
subsys_kl_get_options:KERNEL LOGGING
SS:539:Test_ID_9:12:k:libklfmt.sl:klfmt:subsys_kl_format: \
subsys_kl_get_options:KERNEL LOGGING
```

注释：本示例的续行符（每行末尾的 \ ）以及后面示例中的续行符仅仅是为了增强可读性。 **nettl** 、 **kl** 和 **netfmt** 并不能理解续行符。

配置文件必须始终 *must always* 包含以下条目。它为格式化程序本身定义子系统；如果文件不包含这个条目，格式化程序 netfmt 将不能正常运行。

```
#
# FORMATTER SUBSYSTEMS
#

SS:127:FORMATTER:12:u:libfmtutil.sl:netfmt: \
subsys_GENERIC_format:subsys_127_get_options:FORMATTER

SS:512:FORMATTER:e:u:libfmtutil.sl:netfmt: \
kl_GENERIC_format:subsys_512_get_options:KL FORMATTER
```

文件

/etc/nettlgen.conf

另请参阅

netfmt(1M)、 nettl(1M)、 kl(1M)、 nettlconf(1M)。

名称

networks - 网络名数据库

说明

/etc/networks 文件将 Internet (IP) 地址与正式的网络名和别名关联。这允许用户通过符号名而不是使用 Internet 地址引用网络。对于每个网络，应当有一行带有如下信息：

<official network name> *<network number>* *<aliases>*

别名是用于识别已知网络的其他名称。例如：

loop 192.46.4 testlan

其中，名为 **loop** 的网络也被称为 **testlan** 。

行不能以空白字符开（制表符或空格字符）。项目通过任意数量的空白字符的组合分隔。 **#** 字符标识注释的开始。搜索文件的例行程序不会解释从 **#** 开始直到行的结束的字符。结尾的空白允许在行的结尾。对于 Internet，此文件通常创建于正式的网络数据库，此数据库在网络信息控制中心 (NIC) 维护，尽管可能需要有关非正式的别名和（或）未知的网络的本地更改保持最新。

网络数字可以在常规的 Internet 点的表示法下，使用 **inet_network()** 来自于 Internet 地址操作库的例行程序（请参阅 *inet*(3N) ）来指定。网络名可能包含除空格、换行或注释字符之外的任意可打印字符。

举例

请参阅 **/etc/networks** 。

作者

networks 由加州大学伯克利分校开发。

文件

/etc/networks

另请参阅

getnetent(3N)。

名称

nisfiles - NIS+ 数据库文件和目录结构

概要

/var/nis

说明

网络信息服务增强版 (NIS+) 使用基于内存的复制数据库。这个数据库使用 **/var/nis** 目录中的一个文件集对稳定存储进行检查点操作，并维护一个事务处理日志。此外， NIS+ 服务器和客户端使用该目录中的文件存储绑定和状态信息。

NIS+ 实现一个建立在安全 RPC 之上的身份验证和授权系统。在这个实现方案中，服务使用一个名为 **cred.org_dir.***domain-name* 的表经授权可访问 NIS+ 命令空间的主体的公有密钥和私有密钥。它在作为 *group* 对象的子域 **groups_dir.***domain-name* 中存储组访问信息。这两个表位于 NIS+ 服务器上的 **/var/nis/***hostname* 目录下的文件中。

与 NIS+ 中的以前版本的网络信息服务不同，这里是从服务器上的 ASCII 文件初将表中的信息最初加载到服务中，然后使用 NIS+ 实用程序 (**nistbladm -D**) 更新。出于归档的考虑，有些站点期望周期性地重新生成 ASCII 文件。为此，应在列出了这些表并根据结果创建了 ASCII 文件的服务器的 *crontab*(1) 中追加一个脚本。

注释：除了 **NIS_COLDSTART** 文件和 **NIS_SHARED_DIRCACHE** 文件， *cp*(1) 、 *mv*(1) 或 *rm*(1) 不能操作其他任何文件。事务处理日志文件保存对所有更改的记录，因此不能独立处理这些文件。

下面描述的文件存储于 **/var/nis** 目录：

NIS_COLDSTART
文件包含需要在启动时预载入 NIS+ 缓存的 NIS+ 目录对象。这个文件通常是在安装 NIS+ 时创建的。请参阅 *nisinit*(1M) 或 *nisclient*(1M) 。

NIS_SHARED_DIRCACHE
文件包含由缓存管理器维护的 NIS+ 绑定的当前缓存。可以使用 *nisshowcache*(1M) 查看其内容。

*hostname***.log**
文件包含由 NIS+ 服务维护的事务处理日志。可以使用 *nislog*(1M) 命令查看其内容。此文件包含空隙。其外观大小可能远大于实际大小。每个服务器只有一个事务处理日志。

*hostname***.dict**
此文件是一个字典， NIS+ 数据库使这个字典来定位其文件。它是由缺省的 NIS+ 数据库包创建的。

*hostname***.dict.log**
这是数据库字典的日志文件。当服务器进行检查点操作 (**nisping -C**) 时，此文件将被删除。

hostname 此目录包含服务器所使用的数据库。

*hostname***/root.object**
在根服务器上，这个文件包含一个目录对象，该目录对象描述命名空间的根。

hostname/**parent.object**

在根服务器上，这个文件包含描述父命名空间的目录对象。这个文件由 *nisinit*(1M) 命令创建。

hostname/*table_name*

对于目录中的每个表，都有具有相同文件名且存储了表的有关信息的文件。如果此目录包含子目录，那么表的数据库存储在文件 *table_name***.***subdirectory*. 中。

hostname/*table_name.log*

此文件包含表 *table_name* 的数据库日志。日志文件在每个数据库中维护单个事务处理的状态。对数据库进行检查点操作后（即完成了对 *hostname*/*table_name* 稳定存储的所有更改）后，此日志文件将被删除。

目前 NIS+ 并不自动执行检查点操作。系统管理员可能想周期性地执行 **nisping**-C （请参阅 *nisping*(1M) ）操作（如每日一次），以对日志文件进行检查点操作。这可以通过 *cron*(1M) 作业来完成，也可以手动完成。

hostname/**root_dir**

在根服务器上，此文件存储与根目录相关联的数据库。它与其他表数据库是相似的。相应的日志文件称为 **root_dir.log** 。

hostname/**cred.org_dir**

此表包含此 NIS+ 域中的主体凭证。

hostname/**groups_dir**

此表包含 NIS+ 授权组访问所需的组授权对象。

hostname/**serving_list**

文件包含这个服务器上 NIS+ 服务器正在服务的所有 NIS+ 目录的列表。从任何 NIS+ 目录对象添加和删除此服务器后，服务器将更新这个文件。

警告

HP-UX 11i v2 是支持 NIS+ 的最后一个 HP-UX 版本。

建议使用 LDAP 替代 NIS+。HP 完全支持基于 LDAP 的行业标准命名服务。

作者

nisfiles 由 Sun Microsystems, Inc. 开发。

另请参阅

cp(1)、 crontab(1)、 mv(1)、 rm(1)、 nis+(1)、 niscat(1)、 nismatch(1)、 nistbladm(1)、 nisclient(1M)、 nisinit(1M)、 nislog(1M)、 nisping(1M)、 nis_db(3N)、 nis_objects(3N)。

名称

nlist、nlist64 - nlist 与 nlist64 结构格式

概要

备注

对于基于 Itanium® 的系统，请参阅 *nlist_ia*(4) 。

对于 PA-RISC 系统，请参阅 *nlist_pa*(4) 。

使用 **uname** 命令确定您的系统类型。在基于 Itanium 的系统上， **uname -m** 返回 **ia64** 。所有其他值表示 PA-RISC 系统。

另请参阅

nlist_ia(4)、 nlist_pa(4)、 uname(1)。

名称

nlist_ia： nlist、nlist64 - nlist 和 nlist64 结构格式

概要

#include <nlist.h>

备注

下面所定义的结构的准确内容，可通过查看 **/usr/include/nlist.h** ，找到最完整的答案。在不同的 HP-UX 实现中略微有所变化。

说明

nlist() 和 **nlist64()** 可用于从对象文件的符号表中提取信息（请参阅 *nlist*(3E) ）。它们基本上是相同的工具，都可以处理 SOM 和 Elf 文件。由于符号表取决于计算机（在 **<a.out.h>** 的每一个实现的副本中进行定义），因此定义头文件 **nlist.h** 来封装差异。

函数 nlist、或者 **nlist()** 、或者 **nlist64()** ，在与相应的 nlist 结构一起使用时，可用于从符号表中提取关于所选符号的某些信息。与每个符号相关联的数据是计算机特定的，因此，函数中只有 **n_name** 字段的的名称和位置由 HP-UX 进行标准化。结构的其余部分至少要包括符号的值和类型。未进行标准化的所有字段名称和含义，将只根据需要而改变。

nlist 结构与 **nlist64** 结构相同，用于源代码兼容。

```
struct nlist64 {
   char    *n_name;
   /* other fields as needed;
     the following are suggested if they apply */
     char                *n_qual;
     unsigned short      n_type;
     unsigned short      n_scope;
     unsigned long       n_info;
     unsigned long long n_value;
     unsigned int        is_elf:1;
     unsigned int        is_32:1;
     unsigned int        reserved1:30;
     unsigned long long reserved2;
     unsigned long long reserved3;
};
```

另请参阅

nlist(3E)、 a.out(4)。

名称

nlist_pa： nlist、nlist64 - nlist 和 nlist64 结构格式

概要

#include <nlist.h>

备注

下面定义的结构的确切内容最好通过 **/usr/include/nlist.h** 找到。它随各种 HP-UX 实现的不同而有所变化。

说明

nlist() 和 **nlist64()** 可以用于从对象文件中的符号表中提取信息（请参阅 *nlist*(3C)）。基本上它们是同一个工具，区别在于 **nlist()** 只能处理 PA-RISC 32 位系统上的 SOM 文件，而 **nlist64()** 可以处理 PA-RISC 32 位或 64 位系统上的 SOM 和 ELF 文件。由于符号表取决于计算机（在 **<a.out.h>** 的每一个实现的副本中进行定义），因此定义头文件 **nlist.h** 来封装差异。

nlist 函数为 **nlist()** 或 **nlist64()**，当与相关的 nlist 结构一起使用时，可以用于提取关于符号表中选定符号的特定信息。与每个符号相关的数据是特定于计算机的，因此只有函数中的 **n_name** 的名称和位置字段通过 HP-UX 进行了标准化。结构的其余部分至少包括了符号的值和类型。未加以标准化的所有字段之名称和含义只有在必要时才会更改。

```
struct nlist {
   char    *n_name;
   /* other fields as needed;
     the following are suggested if they apply */
      char                *n_qual;
      unsigned short   n_type;
      unsigned short   n_scope;
      unsigned int       n_info;
      unsigned long     n_value;
};

struct nlist64 {
   char    *n_name;
   /* other fields as needed;
     the following are suggested if they apply */
      char                     *n_qual;
      unsigned short        n_type;
      unsigned short        n_scope;
      unsigned long          n_info;
      unsigned long long  n_value;
      unsigned int            is_elf:1;
      unsigned int            is_32:1;
      unsigned int            reserved1:30;
```

```
        unsigned long long reserved2;
        unsigned long long reserved3;
    };
```

另请参阅

nlist(3C)、 a.out(4)。

名称

nlspath - NLSPATH 配置文件

说明

文件 **/etc/default/nlspath** 允许超级用户通过环境变量 **NLSPATH** 限制由其他用户设置的路径以查找 **setuid** 或 **setgid** 根程序的消息清单。

此文件仅包含一个条目，格式如下所示：

NLSPATH=*pseudo-pathname***:***pseudo-pathname***:***...*

pseudo-pathnames 必须以冒号分隔。在配置文件 **/etc/default/nlspath** 和环境变量 **NLSPATH** 中都可用的路径被被考虑用来查找消息清单文件。不应直接编辑 **/etc/default/nlspath** 文件。相反，应使用 **chnlspath** 命令修改此文件的内容。有关详细信息，请参阅 *chnlspath*(1M) 。

如果 **/etc/default/nlspath** 包含 **NLSPATH=*** ，则文件是在兼容模式下。这种情况下，所有 **setuid** 和 **setgid** 根程序直接使用 **NLSPATH** 环境变量查找消息清单。此配置文件仅用于支持提供 **setuid** 或 **setgid** 根程序的向后兼容，这些根程序依赖于环境变量 **NLSPATH** 。新的 **setuid** 或 **setgid** 根程序不应当依赖于环境变量 **NLSPATH** 和配置文件 **/etc/default/nlspath** 。有关详细信息，请参阅 *catopen*(3C) 。

警告

配置文件必须是根用户所有，而且不应设置组和其他用户的写入权限。超级用户不应为配置文件中提及路径的组和其他用户提供写入权限。

文件

/etc/default/nlspath

另请参阅

catopen(3C)、 chnlspath(1M)、 environ(5)。

名称

nsswitch.conf、switch - 名称服务交换配置文件

概要

/etc/nsswitch.conf

说明

操作系统使用很多信息的“数据库”，这些信息与主机、用户 (**passwd**) 、组等相关。这些数据可以取自各种来源：例如，主机名和地址，可以在 **/etc/hosts** 、NIS、NIS+、LDAP 或 DNS 中找到。每个数据库可以使用一个或多个来源；这些来源及它们的查找顺序在 **/etc/nsswitch.conf** 文件中指定。

下列数据库使用数据服务交换文件：

数据库	使用者
aliases	**sendmail**
automount	**automount**
group	**getgrnam()**
hosts	**gethostbyname()**
netgroup	**innetgr()**
networks	**getnetbyname()**
passwd	**getpwnam()** 、 **getspnam()**
protocols	**getprotobyname()**
publickey	**getpublickey()** 、 **secure_rpc()**
rpc	**getrpcbyname()**
sendmailvars	**sendmail**
services	**getservbyname()**
ipnodes	**getipnodebyname()**

可能使用的来源如下：

来源	使用
files	**/etc/hosts** 、 **/etc/passwd** 等
nis	NIS (YP)
nisplus	NIS+
ldap	LDAP 目录服务器
dns	仅对 **hosts** 、 **ipnodes** 有效；使用 Internet 域名服务。
compat	仅对 **passwd** 和 **group** 有效；实现 **+** 和 **-** 。（请参阅下面的使用 +/- 语法交互）

在 **/etc/nsswitch.conf** 中有每个数据库的条目。通常这些条目很简单，例如“protocols:files”或“networks:files nisplus”。然而，当指定了多个来源时，有时需要准确定义环境，在此环境下尝试每个来源。一个来源可以返回下列代码之一：

状态	含义
SUCCESS	请求的数据库条目已找到
UNAVAIL	来源没有响应或已损坏
NOTFOUND	来源响应“无此条目”
TRYAGAIN	来源正忙，可响应以重试

对于每个状态代码，可能有两种操作：

Action	*Meaning*
continue	尝试列表中的下一个来源
return	立刻返回

一个条目的完整语法为

<entry> ::= <database> ":" [<source> [<criteria>]]* <source>
<criteria> ::= "[" <criterion>+ "]"
<criterion> ::= <status> "=" <action>
<status> ::= "success" | "notfound" | "unavail" | "tryagain"
<action> ::= "return" | "continue"

每个条目占据文件中的一行。忽略空行或以空格字符开头的行。一行中 **#** 字符之后的所有内容也将忽略； **#** 字符可以在一行中的任意位置开始，用于开始注释。 *database* 和 *source* 名称区分大小写，但是 *action* 和 *status* 名称不区分大小写。

缺省条件是对除了 **SUCCESS;** 之外的所有状态继续操作；也就是说， **[SUCCESS=return NOTFOUND=continue UNAVAIL=continue TRYAGAIN=continue]** 。

在缺省情况，或明确指定的情况下，条目中最后一个来源之后的条件没有意义；并且该条件将被忽略，因为不管该来源返回什么状态代码，该操作总是返回到调用处。

与 **Netconfig** 交互

为了确保它们根据 **inet** 系列条目返回一致的结果， **gethostbyname()** 、 **getservbyname()** 和 **netdir_getbyname()** 函数都实现相同的内部交换库功能。这些函数在系统级内为 **hosts** 和 **services** 获取基于 **inet** 系列条目在 **netconfig()** 中的来源查找策略。对于 **services** 和 **hosts** ，只有在最后一列中代表 **nametoaddr** 库的 **-** 才受支持。

与 **NIS+ YP** 兼容模式交互

NIS+ 服务器可以在“YP 兼容模式”下运行，在该模式下处理 NIS (YP) 请求以及 NIS+ 请求。在这种情况下，客户端从“nis”源获得与从“nisplus”相同的结果；但是推荐使用“nisplus”而不使用“nis”。

与 **DNS** 转发模式下的 **NIS (YP)** 服务器交互

NIS (YP) 服务器可以在“DNS 转发模式”下运行，在该模式下，服务器转发主机名和地址不在其数据库中的 DNS 请求。在这种情况下，将“nis”指定为“主机”的来源就足够进行 DNS 查找了；不必明确地将“dns”指定为来源。

在“YP 兼容模式”下运行的 NIS+ 服务器也可以在“DNS 转发模式”下运行（请参阅 *rpc.nisd*(1M) ）。转发只

对来自其 YP 客户端的请求有效；应适当配置这些客户端上的“主机”策略。

与 +/- 语法交互

HP-UX 10.30 之前的版本没有对口令和组的名称服务交换支持，但确实允许用户进行一些策略控制。在 **/etc/passwd** 中可以有 **+***user* （包括从 NIS *passwd.byname* 中指定的用户）、 **-***user* （不包括指定的用户）和 **+** （包括除了从 NIS *passwd.byname* 中排除的用户之外的所有内容）形式的条目。所需的行为方式通常是“文件中的全部内容，然后是 NIS 中的全部内容”，这是通过将一个 **+** 放在 **/etc/passwd**末尾来表示的。交互文件提供了另一种方法解决（"passwd:files nis"）这种在 **/etc/passwd** 中不需要 **+** 的情况。

如果这样还不够，“compat”源提供了完整的 **+/-** 语义。它为 **getpwnam()** 函数读取 **/etc/passwd** ，然后如果找到 **+/-** 条目，则调用适当的来源。缺省情况下，来源是“nis”，但是此来源可能会通过将“nisplus”指定为 pseudo 数据库 **passwd_compat** 的来源而被覆盖。

compat 源也可以为 **group** 提供完整的 **+/-** 语义；相关的 pseudo 数据库是 **group_compat** 。

这些库函数包含编译进来的缺省条目，这些缺省条目当在 **nsswitch.conf** 中不存在适当的条目时，或在句法上不正确时使用。这些条目如下所示：

passwd:	**files nis**
group:	**files nis**
hosts:	**dns [NOTFOUND=return] nis [NOTFOUND=return] files**
networks:	**nis [NOTFOUND=return] files**
protocols:	**nis [NOTFOUND=return] files**
rpc:	**nis [NOTFOUND=return] files**
publickey:	**nis [NOTFOUND=return] files**
netgroup:	**nis [NOTFOUND=return] files**
automount:	**files nis**
aliases:	**files nis**
services:	**nis [NOTFOUND=return] files**
ipnodes:	**dns [NOTFOUND = return] files**

有用的配置

所有数据库中编译进来的缺省条目使用 NIS (YP) 作为企业级的名称服务，并且与下述文件缺省配置中的条目相同：

passwd:	**files nis**
group:	**files nis**
hosts:	**nis [NOTFOUND=return] files**
networks:	**nis [NOTFOUND=return] files**
protocols:	**nis [NOTFOUND=return] files**
rpc:	**nis [NOTFOUND=return] files**
publickey:	**nis [NOTFOUND=return] files**

netgroup:	**nis**
automount:	**files nis**
aliases:	**files nis**
services:	**files nis**
sendmailvars:	**files**
ipnodes:	**files**

策略 **nis [NOTFOUND=return] files** 表示“如果 **nis** 是 **UNAVAIL** ，则继续进行到 **files** ；如果 **nis** 返回 **NOTFOUND** ，则返回到调用者；否则，将 **nis** 当作信息的授权源，并且当且仅当 **nis** 失败时尝试 **files** ”。

如果需要口令和组与 **+/-** 语法兼容，则为 **passwd** 和 **group** 将条目简单地修改为：

passwd:	**compat**
group:	**compat**

如果 NIS+ 是企业级名称服务，则客户机上每个数据库的缺省配置应修改为使用 **nisplus** 而不是 **nis** 。文件 **/etc/nsswitch.nisplus** 包含一个示例配置，该配置应复制到 **/etc/nsswitch.conf** 来设置本策略。

如果需要将 **+/-** 语法与 **nisplus**联合使用，则使用以下四个条目：

passwd:	**compat**
passwd_compat:	**nisplus**
group:	**compat**
group_compat:	**nisplus**

要从 Internet 域名服务获得没有在企业级名称服务中列出的主机信息， NIS+ 使用该配置并设置文件 **/etc/resolv.conf** 。有关详细信息，请参阅 *resolver*(4) 。

hosts:	**nisplus dns [NOTFOUND=return] files**
ipnodes:	**dns [NOTFOUND=return] files**

文件 **/etc/nsswitch.ldap** 包含可以复制到 **/etc/nsswitch.conf** 以设置 LDAP 策略的示例配置。如果需要 **+/-** 网络组语法（用于 **nis** 和 **nisplus** 定义的访问控制），则管理员需要配置 **libpam_authz.1** （在 **/etc/pam.conf** 文件中）。请参阅 *ldapux*(5) 联机帮助页获得关于 LDAP-UX 的详细信息， *pam_authz*(5) 联机帮助页获得关于 **libpam_authz.1** 的详细信息，以及 *passwd*(4) 获得关于 **+/-** 网络组语法的详细信息。 *ldapux*(5) 和 *pam_authz*(5) 联机帮助页位于 LDAP-UX 集成产品中。

枚举 -- getXXXent( )

很多数据库都有枚举功能： **passwd** 有 **getpwent()**， **hosts** 有 **gethostent()**，等等。当仅有的来源是 **files** 时是合理的，但是它对于包含大量条目，而对于多个来源来说要少很多的层次结构源没有意义。数据库仍提供接口，且执行过程力求提供合理的结果，但是返回的数据可能会不完全（ **dns** 源不支持 **hosts** 枚举）、不一致（如果使用了多个来源）、格式是意想不到的风格（针对名称不规范以及有三个别名的主机， **nisplus** 源会返回四个主机名，并且他们可能不连贯）或者代价太高（列举 5000 个用户 **passwd** 的数据库可能是一个可怕的想法）。此外，同一进程中使用相同 reentrant 枚举函数（支持 **get***XXX***ent_r()** ）的多个线程分享相同的枚举位置；如果他们交叉调

用，则他们将枚举同一数据库的不相交子集。

总之，不赞成使用枚举函数。在 **passwd** 和 **group** 的情况下，有时可能使用 **fgetgrent()**、 **fgetpwent()** 和 **fgetspent()** 更合适（请分别参阅 *getgrent*(3C) 和 *getpwent*(3C)），它们只使用 **files** 源。

警告

使用 **nsswitch.conf()** 的进程中，整个文件只读取一次。如果该文件之后发生了更改，该进程会继续使用旧的配置。

使用 **get***XX***by***YY***()** 函数的程序不能动态链接，因为执行该函数需要动态链接功能在运行的时候访问服务共享对象 **/usr/lib/nss_SSS.sl.1** 。

强烈建议不要将 **nis** 和 **nisplus** 都用作同一数据库的来源，因为这两个名称服务都希望保存类似的信息，并且在数据库上的查找可能产生不同的结果，其结果取决于请求时哪个名称服务正在操作。

如果来源和数据库名称拼写错误，则会被视作其名称为非法（很可能不存在）。

下列功能 *not* 不使用交换文件： **fgetgrent()** 、 **fgetpwent()** 、 **fgetspent()** 、 **getpw()** 和 **putpwent()** 。

函数 **getipnodebyname()** 和 **getipnodebyaddr()** 通过 **libc.2** 介绍并且在 **libc.1** 中找不到。

与 **libc.1** 链接的应用程序会为 **NOTFOUND** 和 **TRYAGAIN** 显示不同的缺省操作。与 **libc.1** 链接的应用程序具有交换文件搜索终止符，其前提是名称服务返回结果 **NOTFOUND** 或 **TRYAGAIN** 。

这可能是现有 **nsswitch.conf** 文件的一个结果，该文件指定了在 *source* 条目之间不包含 *criterion* 的名称服务查找条件。

例如： **hosts: dns files**

对于与 **libc.1** 链接的应用，返回到文件只会发生在 **DNS** 返回 **UNAVAIL** 时。对于所有其他应用，返回到文件只发生在 **DNS** 返回 **SUCCESS** 时。

对于与 **libc.1** 链接的应用以及其他有相同行为的应用，必须指定 *criterion* 来定义 *source* 。

对于 **libc.1** 行为：

hosts: dns [NOTFOUND=return TRYAGAIN=return] files

对于缺省的系统行为：

hosts: dns [NOTFOUND=continue TRYAGAIN=continue] files

作者

nsswitch.conf 由 Sun Microsystems, Inc. 开发。

文件

命名为 SSS 的来源由名称为 **nss_SSS.1** 的共享对象实现，该对象位于 **/usr/lib** 中。

/etc/nsswitch.conf 配置文件

/usr/lib/nss_compat.1	实现“compat”源
/usr/lib/nss_dns.1	实现“dns”源
/usr/lib/nss_files.1	实现“files”源
/usr/lib/nss_nis.1	实现“nis”源
/usr/lib/nss_nisplus.1	实现“nisplus”源
/usr/lib/nss_ldap.1	实现“ldap”源
/etc/netconfig	**netdir()** 函数的配置文件，可将主机/服务策略重定向至交换文件
/etc/nsswitch.files	只使用“files”的示例配置文件
/etc/nsswitch.nis	只使用“files”和“nis”的示例配置文件
/etc/nsswitch.nisplus	使用“files”和“nisplus”的示例配置文件
/etc/nsswitch.ldap	使用“files”和“ldap”的示例配置文件

另请参阅

nis+(1)、 automount(1M)、 rpc.nisd(1M)、 sendmail(1M)、 getgrent(3C)、 gethostent(3N)、 getnetent(3N)、 getnetgrent(3C)、 getprotoent(3N)、 getpublickey(3N)、 getpwent(3C)、 getrpcent(3C)、 getservent(3N)、 netdir(3N)、 secure_rpc(3N)、 netconfig(4)、 resolver(4)、 ypfiles(4)。

LDAP-UX 集成产品中的 ldapux(5) 和 pam_authz(5)。

名称

pam.conf - 可插拔身份验证模块的配置文件

概要

/etc/pam.conf

说明

pam.conf 是可插拔身份验证模块体系结构 PAM 的配置文件。 PAM 模块提供一个或多达四种可能的服务功能：身份验证、 帐户管理、 会话管理和 口令管理。

身份验证服务模块提供对用户进行验证和设置用户凭证的功能。帐户管理模块提供确定当前用户帐户是否有效的功能。包括口令和帐户有效期检查，以及访问时间段限制验证。会话管理模块提供设置和终止登录会话的功能。口令管理模块提供更改用户身份验证标记或口令的功能。

简化的 **pam.conf** 配置文件

pam.conf 文件包含服务列表。每项服务对应于一个相应的服务模块。当提出服务需求时，相应的模块被激活。每个服务条目格式如下：

service_name module_type control_flag module_path options

下面是一个 **pam.conf** 配置文件的示例，这个文件支持身份验证、帐户管理、登录时间管理和口令管理等模块。注意有些服务强制使用 **pam_hpsec** 。有关详细信息，请参阅 *pam_hpsec*(5) 。

```
login    auth     required  libpam_hpsec.so.1 debug
login    auth     required  libpam_unix.so.1  debug
login    session  required  libpam_hpsec.so.1
login    session  required  libpam_unix.so.1
login    account  required  libpam_unix.so.1
dtlogin  auth     required  libpam_hpsec.so.1
dtlogin  auth     required  libpam_unix.so.1
dtlogin  session  required  libpam_hpsec.so.1
dtlogin  session  required  libpam_unix.so.1
other    auth     required  libpam_unix.so.1
other    account  required  libpam_unix.so.1
other    session  required  libpam_unix.so.1
other    password required  libpam_unix.so.1
```

service_name *service_name* 表示服务名称（如 **login** 或 **dtlogin** ）。关键字 **other** 指明所有其他未指定的应用程序应使用的模块。如果同一 *module_type* 的所有服务具有相同的需求，则也可使用 **other** 关键词。在上面的示例中，由于所有的服务使用相同的帐户管理模块，因此可用一个 **other** 行来替换。

module_type *module_type* 表示服务模块名称：身份验证 (*auth*) 、帐户管理 (*account*) 、登录时间管理 (*session*) 或口令管理 (*password*) 。

control_flag *control_flag* 字段确定堆栈的行为，有关细节将在下面讨论。

module_path *module_path* 字段指定实现服务功能的共享库对象的路径名。如果不是绝对路径，则假定路径相对于 **/usr/lib/security/$ISA/** 。 **$ISA** （即指令集体系结构）标记将由 PAM 引擎 (libpam) 进行替换。对基于 Itanium 的 32 位模块用 **hpux32** 替换，对 PA-RISC 32 位模块来说为空，对基于 Itanium 的 64 位模块，用 **hpux64** 替换，而对 PA-RISC 64 位模块则用 **pa20_64** 替换。为了保持对 PA-RISC 库命名惯例向后兼容， **/usr/lib/security/** 和 **/usr/lib/security/pa20_64** 中提供了适当的链接。例如： **/usr/lib/security/libpam_unix.so.1 -> ./libpam_unix.1** 如果用户定义模块在 **/etc/pam.conf** 或 **/etc/pam_user.conf** 中进行规定，则必须遵守上述惯例以创建指向 PA-RISC 模块的符号链接。为了有助于减少任何将来 **/etc/pam.conf** 文件格式发生变化的影响，唯一支持的解析 **/etc/pam.conf** 文件的方法通过 PAM 库接口进行。这些接口将对保留标记（如 **$ISA** ）进行必要的扩展，此操作对用户是透明的。

options PAM结构层使用 *options* 字段来向模块传送特定模块的选项。由模块来对选项进行分析和解释。模块可利用这个字段来启动调试或传递任何模块特定的参数，如 TIMEOUT 值。也可用于支持统一登录。各模块所能支持选项在相应手册页中介绍。如 *pam_unix*(5) 中列出了 UNIX 模块可接受的选项。

使用堆栈集成的多身份验证服务

当相同 *module_type* 的服务名的定义多于一次，则称这类服务为 *stacked* 。在 *module_path* 中引用的此项服务的每个模块按照配置文件中的顺序进行处理。 *control_flag* 字段规定了模块的延续和失效的意义，其取值可以是 **required** 、 **optional** 或 **sufficient** 。

每个 PAM 模块返回一个状态，这个状态指出此模块是否对请求的操作作出批准或无意见。若模块操作成功但无操作结果，则忽略此模块相应的控制标志。

PAM 框架处理堆栈中的每个模块。若堆栈中所有 **required** 模块都操作成功，则返回成功标志（此时忽略 optional 和 **sufficient** 错误值）。若一个或多个 **required** 模块操作失败，则返回第一个操作失败的 **required** 模块的错误值。

若堆栈中的服务模块没有被设定为 **required** ，则 PAM 框架要求至少有一个 **optional** 或 **sufficient** 模块操作成功。若全部操作都失败，则返回堆栈中第一个服务模块的错误值。

上面所述只有一个例外，是由 **sufficient** 标志引起。若设定为 **sufficient** 的服务模块操作成功， PAM 框架立即向应用程序返回操作成功标志（堆栈中所有后续模块都被忽略，包括设定为 **required** 的模块），其前提是前面所有的 **required** 模块也都操作成功。若前面的 **required** 模块操作失败，则返回该模块的错误值。

若模块不存在或打不开，则 **LOG_CRIT** 级别的 *syslog*(3C) 将对错误进行记录，并且 PAM 框架向应用程序返回 **PAM_OPEN_ERR** 错误标志。

下面是一个对 **login** 和 **dtlogin** 服务进行堆栈操作的配置文件示例。

```
login   auth  required   libpam_hpsec.so.1 debug
login   auth  required   libpam_unix.so.1  debug
login   auth  optional   libpam_inhouse.so.1
```

```
dtlogin auth required   libpam_hpsec.so.1  debug
dtlogin auth sufficient libpam_unix.so.1   debug
dtlogin auth required   libpam_inhouse.so.1
```

在请求 **login** 操作的情况下，用户由 hpsec、 UNIX 和 inhouse 身份验证模块进行认证。 *control_flag* 的关键字 **required** 表示用户只有通过 hpsec 和 UNIX 服务模块身份验证后才能进行登录。当 *control_flag* 字段为关键字 **optional** 时，Inhouse 身份验证是可选的。 即使 inhouse 身份验证失败用户也能进行登录。

在请求 **dtlogin** 操作的情况下， *control_flag* 的关键字 **sufficient** 表示用户通过 UNIX 身份验证后，PAM 应向 **dtlogin** 返回成功标志。inhouse 身份验证模块（堆栈中的下一个模块）仅在 UNIX 身份验证失败后才激活。

按用户配置

pam.conf 包含了配置系统中所有用户的信息。但有时有必要按用户进行配置。用户策略的定义通过一个名称为 **libpam_updbe.so.1** 的特定模块进行。这个模块读取文件名为 **/etc/pam_user.conf** 的文件，从中获取用户配置信息。

下面是使用 **libpam_updbe.so.1** 模块的示例配置文件 (**/etc/pam.conf**) 。

```
login   auth     required   libpam_hpsec.so.1
login   auth     required   libpam_updbe.so.1
login   auth     required   libpam_unix.so.1
su      auth     required   libpam_hpsec.so.1
su      auth     required   libpam_updbe.so.1
su      auth     required   libpam_unix.so.1
OTHER   auth     required   libpam_unix.so.1

login   password required   libpam_hpsec.so.1
login   password required   libpam_updbe.so.1
login   password required   libpam_unix.so.1
passwd  password required   libpam_hpsec.so.1
passwd  password required   libpam_updbe.so.1
passwd  password required   libpam_unix.so.1
OTHER   password required   libpam_unix.so.1
```

libpam_updbe.so.1 模块搜索配置文件 **/etc/pam_user.conf** 并读取与当前用户的登录名相关的配置信息。若配置文件 **pam_user.conf** 中没有当前用户的配置信息，则 PAM 框架忽略包含 **libpam_updbe.so.1** 的一行。配置文件 pam.conf 应用于那些 **pam_user.conf** 中没有配置的用户。

注释

若在服务条目中发现由于非法的 *service_name* 、 *module_type* 或 *control_flag* 引起的错误，则忽略此条目。若对于给定的 *module_type* 没有合法的条目，则 PAM 框架向应用程序返回出错标志。

举例

下面是配置文件 **pam.conf** 的示例：以符号 **#** 开始的行是注释行，因此程序处理时予以忽略。

```
#
# PAM configuration
#
# Authentication management for login service is stacked.
# Both UNIX and inhouse authentication functions are invoked,
# in addition to hpsec authentication functions.
login   auth    required    libpam_hpsec.so.1
login   auth    required    libpam_unix.so.1
login   auth    required    libpam_inhouse.so.1  try_first_pass
dtlogin auth    required    libpam_hpsec.so.1
dtlogin auth    required    libpam_unix.so.1
dtlogin auth    required    libpam_inhouse.so.1  try_first_pass
#
# Other services use UNIX authentication
other   auth     required     libpam_unix.so.1
#
# Account management for login service is stacked.
# hpsec and UNIX account management are required;
# inhouse account management is optional
login   account  required     libpam_hpsec.so.1
login   account  required     libpam_unix.so.1
login   account  optional     libpam_inhouse.so.1
dtlogin account  required     libpam_hpsec.so.1
dtlogin account  required     libpam_unix.so.1
dtlogin account  optional     libpam_inhouse.so.1
#
# Other services use UNIX account management
other   account  required     libpam_unix.so.1
#
# Session management for login service is stacked.
# hpsec and UNIX account management are required;
login   session  required     libpam_hpsec.so.1
login   session  required     libpam_unix.so.1
dtlogin session  required     libpam_hpsec.so.1
dtlogin session  required     libpam_unix.so.1
#
# Other services use UNIX session management
```

```
other    session   required      libpam_unix.so.1
#
# Password management
other    password  required      libpam_unix.so.1
```

下面是一个使用 **libpam_updbe.so.1** 模块对单个用户进行配置的 **pam.conf** 配置文件的示例。以符号 **#** 开始的行是注释行，因此程序处理时予以忽略。

```
#
# PAM configuration
#
# Authentication management for login service is stacked.
# Both UNIX and inhouse authentication functions are invoked,
# in addition to hpsec authentication functions.
login    auth     required    libpam_hpsec.so.1
login    auth     required    libpam_updbe.so.1
login    auth     required    libpam_unix.so.1
login    auth     required    libpam_inhouse.so.1   try_first_pass
dtlogin  auth     required    libpam_hpsec.so.1
dtlogin  auth     required    libpam_updbe.so.1
dtlogin  auth     required    libpam_unix.so.1
dtlogin  auth     required    libpam_inhouse.so.1   try_first_pass
#
# Other services use UNIX authentication
other    auth      required      pam_unix.so.1
#
# Account management for login service is stacked.
# hpsec and UNIX account management are required;
# inhouse account management is optional
login    account  required      libpam_hpsec.so.1
login    account  required      libpam_unix.so.1
login    account  optional      libpam_inhouse.so.1
dtlogin  account  required      libpam_hpsec.so.1
dtlogin  account  required      libpam_unix.so.1
dtlogin  account  optional      libpam_inhouse.so.1
other    account  required      libpam_unix.so.1
#
# Session management for login service is stacked.
# hpsec and UNIX account management are required
login    session   required      libpam_hpsec.so.1
```

```
login    session  required    libpam_unix.so.1
login    session  optional    libpam_inhouse.so.1
dtlogin  session  required    libpam_hpsec.so.1
dtlogin  session  required    libpam_unix.so.1
dtlogin  session  optional    libpam_inhouse.so.1
#
# Other services use UNIX session management
other    session  required    libpam_unix.so.1
#
# Password management
passwd   password required    libpam_hpsec.so.1
passwd   password required    libpam_updbe.so.1
passwd   password required    libpam_unix.so.1
other    password required    libpam_unix.so.1
```

实用程序和文件

已知的使用 PAM 的实用程序列表包括: **login** 、 **passwd** 、 **su** 、 **dtlogin** 、 **ftp** 、 **remsh/rexec** 和 **ssh** 。

PAM 配置文件既不规定服务特定模块的名称，也不规定其地址。然而有以下惯例:

/usr/lib/security/$ISA/libpam_*service_name*.so.1
实现各种特定身份验证服务的功能。

/etc/pam.conf
配置文件。

/usr/lib/hpux32/libpam.so.1
在基于 Itanium 的系统中实现 32 位 PAM 框架库。

/usr/lib/hpux64/libpam.so.1
在基于 Itanium 的系统中实现 64 位 PAM 框架库。

/usr/lib/libpam.1
在 PA-RISC 系统中实现 32 位 PAM 框架库。

/usr/lib/pa20_64/libpam.1
在 PA-RISC 系统中实现 64 位 PAM 框架库。

另请参阅

dtlogin(1)、 login(1)、 passwd(1)、 su(1)、 pam(3)、 pam_hpsec(5)。

名称

pam_user.conf - 可插拔身份验证模块的用户配置文件

概要

/etc/pam_user.conf

说明

pam_user.conf 是可插拔身份验证模块体系结构或 PAM 的用户配置文件。它不是设计用于替换 PAM 系统配置文件 **pam.conf** 。要使 PAM 恰当地工作，则 **pam.conf** 是必需的（请参阅 *pam.conf*(4)）。 **pam_user.conf** 是可选的。它只在需要用户标准配置时使用。它主要基于用户标准指定了服务模块使用的 *options* 。

pam.conf 中定义的 *options* 表示没有在 **pam_user.conf** 中配置的用户的缺省设置，或者如果该模块类型没有为某些用户进行配置。要使 **pam_user.conf** 中的配置生效，则 **pam.conf** 需要配置服务模块 **libpam_updbe** （请参阅 *pam.conf*(4)）。

简化的 PAM_USER.BONF 配置文件

pam_user.conf 文件包含一个登录名列表。每个登录名以指定或未指定的选项与相应服务模块配对。每个条目具有以下格式:

login_name module_type module_path options

下面是 **pam_user.conf** 配置文件的一个例子。

```
tom     auth     /usr/lib/security/$ISA/libpam_unix.so.1   debug use_psd
tom     auth     /usr/lib/security/$ISA/libpam_dce.so.1    use_first_pass
tom     account  /usr/lib/security/$ISA/libpam_unix.so.1   use_psd
tom     account  /usr/lib/security/$ISA/libpam_dce.so.1    try_first_pass

susan   auth     /usr/lib/security/$ISA/libpam_unix.so.1
susan   auth     /usr/lib/security/$ISA/libpam_dce.so.1    try_first_pass
```

login_name 表示用户的登录名（例如，tom、susan）。关于 *module_type* 、 *module_path* 和 *options* 的详细信息，请参阅 *pam.conf*(4) 。

第一个条目表示当为 *tom* 调用 UNIX 身份验证时，会使用 *options* “debug”和“use_psd”。第二个条目表示当为 *tom* 调用 DCE 身份验证时，会使用 *option* “use_first_pass”。没有为 *tom* 配置模块类型“password”，因此， **/etc/pam.conf** 选项会生效。对那些没有配置的用户，会应用 **/etc/pam.conf** 选项。

注释

如果由于 *login_name* 或 *module_type* 无效而在条目中发现错误，那么该条目被忽略。如果对给定的 *module_type* 没有有效的条目，则 PAM 框架忽略 **pam_user.conf** 并读取 **pam.conf** 中的配置。

举例

下面是一个 **pam_user.conf** 配置文件的示例。以 **#** 符号开头的行被看作注释，因此被忽略。

```
#
# PAM user configuration
```

```
#
# Authentication management
john    auth    /usr/lib/security/$ISA/libpam_unix.so.1
john    auth    /usr/lib/security/$ISA/libpam_inhouse.so.1   try_first_pass

david   auth    /usr/lib/security/$ISA/libpam_unix.so.1      use_psd
david   auth    /usr/lib/security/$ISA/libpam_inhouse.so.1   try_first_pass

susan   auth    /usr/lib/security/$ISA/libpam_unix.so.1      use_psd
susan   auth    /usr/lib/security/$ISA/libpam_inhouse.so.1   try_first_pass

# Password management
john    password /usr/lib/security/$ISA/libpam_unix.so.1
david   password /usr/lib/security/$ISA/libpam_unix.so.1     use_psd
susan   password /usr/lib/security/$ISA/libpam_unix.so.1     use_psd
```

另请参阅

pam(3)、 pam.conf(4)。

名称

passwd - 口令文件

概要

#include <pwd.h>

说明

/etc/passwd 针对每个用户包含以下信息：

- 登录名称
- 加密口令
- 数字用户 ID
- 数字组 ID
- 保留的 gecos ID
- 初始工作目录
- 用作 Shell 的程序

这是一个 ASCII 文件。每个用户条目中每个字段与下一个字段以冒号间隔。每个用户与下一个用户以换行符分隔。这个文件保存在 **/etc** 目录中。它可以有并且确实有常规的读取权限，而且可以用于将数字用户 ID 映射至姓名等情况。

getpwent(3C) 返回一个指针，指向在 **<pwd.h>** 中声明的用户条目 passwd 结构

登录名称必须以字母字符开头，并且只能包含字母、数字和下划线字符。如果登录目录为空，缺省情况用户将被置于 **/** 。如果登录 Shell 为空，则使用 **/usr/bin/sh** 。

建议用户 ID 和组 ID 不要使用 0–99 范围内的数字，以便可分配给系统软件的 ID 不会产生冲突。

gecos 字段可能包含以下标识：用户全名，办公室地点，分机和家庭电话。 gecos 字段可以通过 **chfn** 命令设置，并通过 **finger** 命令显示（请参阅 *chfn*(1) 和 *finger*(1) ）。这两个命令假定该字段中的信息是按上述次序排列。用户真名的一部分在 gecos 字段中能表示为一个 **&** 字符，一些实用程序（包括 **finger** ）通过登录名替换该字符和将登录名称的第一个字母转换成大写来进行扩展。

口令字段

在一个非映像标准系统中，所有口令字段都包含了实际加密口令。在映像标准系统中，所有口令字段都包含一个“x”，而实际加密口令驻留在 **/etc/shadow** 中。在一个信任系统中，所有口令字段都包含一个 '*' 字符，实际加密口令保存在受保护口令数据库中。

以下口令字段的说明仅适用于包含实际加密口令的 **/etc/passwd** 中的条目的口令字段。有关驻留在 **/etc/shadow** 中的加密口令的详细信息，请参阅 *shadow*(4) 联机帮助页，有关驻留在信任系统中的加密口令的详细信息，请参阅该页中安全功能一节。

如果口令字段为空，则登录时无口令或不要求口令。该字段由带有可选口令时限子字段的加密口令组成。

加密口令由 13 个字符构成，均选自如下所述的 64 个字符的“数字”集中。若口令字段中输入了非数字集中的字符（例如 * ），登录将无法进行。

用于表示“数字”的字符是 **.** 代表 0， **/** 代表 1， **0** 至 **9** 代表 2 至 11， **A** 至 **Z** 代表 12 至 37， **a** 至 **z** 代表 38 至 63。

口令 aging 设定为对于特殊用户有效，在口令文件中，特殊用户的加密口令紧跟着一个逗号和一个取自上述字符集的字符组成的非空字符串。（这类字符串必须首先由一个超级用户引入）。这个字符串定义了用于实现口令 aging 的 "age"。

UNIX 保留了具有一个基本日期格式 1970 年 1 月 1 日星期四的内部时间戳。因此，passwd 将 GMT 星期四 00:00 点作为一周的开始。

age 的第一个字符， *M* ，表示口令保持有效的最大周数。口令过期仍尝试登录的用户，将被强制提供一个新的口令。下一个字符， *m* ，表示口令能更改之前必须达到的最少时间，以周为单位。剩余的两个字符定义了口令最后更改的星期（空白字符串等同于 0）。字符 *M* 和 *m* 具有 0 至 63 范围的数值，取值对应于上述的 64 字符的“数字”集。

如果 $m = M = 0$ （派生自字符串 **.** 或 **..** ），用户将被强制在下次登录时更改口令（"age" 将从口令文件的用户条目中消失）。如果 $m > M$ （例如，以字符串 **./** 表示），则只有超级用户（非普通用户）可以更改口令。建议任何时候都不禁止用户更改口令。

安全功能

本节仅适用于信任系统。在一个信任系统中，口令字段缺省始终包含 * 字符。口令和 aging 信息是受保护口令数据库中的替代部分。

在信任系统中，每个用户的加密口令保存在文件 **/tcb/files/auth/** *c* **/user_name** 中（此处字符 *c* 是 *user_name* 中的首字母）。口令信息文件对于公众是不可访问的。加密口令可以超过 13 个字符。例如，用户 **david** 的口令文件保存为 **/tcb/files/auth/d/david** 。除口令外，文件 **/tcb/files/auth/*/*** 中的用户配置文件也包括许多其他字段：

- 数字审计 ID
- 数字审计标志

比如 **/etc/passwd** ，是一个 ASCII 文件。每个用户条目中的字段均以冒号间隔。有关详细信息，请参考 *authcap*(4) 和 *prpwd*(4) 。包含于 **/tcb/files/auth/*/*** 文件中的口令优先于那些包含于 **/etc/passwd** 文件加密口令字段中的口令。使用该文件中的加密口令进行用户验证。有关口令 aging 机制的说明，请参阅 *passwd*(1) 中的 **SECURITY FEATURES** 一节。

有关口令和转换为信任系统的详细信息，请参阅《管理系统和工作组》和 *sam*(1M) 。

网络功能

NIS

文件 **passwd** 可以包含以一个加号 **(+)** 或减号 **(-)** 开始第一列的条目。这类行用于访问网络信息系统数据库。以一个加号 **(+)** 起始的行，用于合并来自网络信息系统的条目。有三种样式的 **+** 条目：

+ 在该点插入网络信息系统口令文件的条目内容；

+*name* 在该点插入来自网络信息系统的 *name* 条目（如果有的话）。

+@_name_ 在该点插入网络组 _name_ 中所有成员的条目。

如果一个 **+** 条目具有非空口令、目录、gecos 或 Shell 字段，则网络信息系统中的相同字段被覆盖。数字用户 ID 和组 ID 字段不能覆盖。

passwd 文件也能包含以一个减号 (**-**) 起始的行，从而拒绝来自网络信息系统的条目。有两种样式的 **-** 条目：

-_name_ 拒绝任何后来的 _name_ 条目（如果有的话）。

-@_name_ 不接受任何后来的网络组 _name_ 中所有成员的条目。

NIS 警告

加号 (**+**) 和减号 (**-**) 特性属于 NIS 功能；因此，如果没有安装 NIS，这些功能将不起作用。此外，这些功能仅在 **/etc/passwd** 中起作用，在系统转换为信任系统后不起作用。当系统已转换为信任系统时，加密口令只能通过受保护口令数据库， **/tcb/files/auth/*/*** ，进行访问。网络信息系统数据库中的任何用户条目在受保护口令数据库中也必须有其关联条目。

−2 uid 被保留用于通过 NFS 访问远程根目录。通常赋予这个 uid 的用户名是 **nobody** 。由于 uid 作为签名值存储，因此以下包含在 **<pwd.h>** 中的定义与用户 **nobody** 相匹配。

UID_NOBODY (-2)

警告

根用户 (uid 0) 的登录 Shell 必须是 **/sbin/sh** ，以此确保系统总能够引导启动。其他 Shell 如 sh、ksh 和 csh 都定位在 **/usr** 目录中，在引导过程的较早阶段可能不被挂接。允许将根用户的登录 Shell 改变为除 **/sbin/sh** 外的其他值，但可能会导致系统不起作用。

保存在 gecos 字段中的信息可能与该字段不支持的用法或将来的使用发生冲突。 gecos 字段保存用户标识信息的用途，在任何一个企业标准中都尚未正式规定。该字段的当前使用派生自它在 Berkeley Software Distribution 中的应用。将来的标准可能为其他目的定义该字段。

注意下列字段都有明确的字数限制进行：

- 登录名称字段不得超过 8 个字符；
- 初始工作目录字段不得超过 63 个字符；
- 程序字段不得超过 44 个字符；
- 如果这些字段超过了以上规定的界限，结果将不可预料。

注意下列字段都有明确的数值限制：

- 用户 ID 是一个 0 到 **UID_MAX**-1 （包括该值）之间的整数值。 -2 可作为一个特例存在。
- 组 ID 是一个 0 到 **UID_MAX**-1 （包括该值）之间的整数值。 -2 可作为一个特例存在。
- 如果任何一个值超出范围，则函数 _getpwent_(3C) 将 ID 值重置为 (**UID_MAX**) 。

举例

映像口令举例

root:x:0:10:System Administrator:/:/sbin/sh

joe:x:100:50:Joe User,Post 4A,12345:/home/joe:/usr/bin/ksh

如果系统已转换为一个映像标准系统，用户 **root** 和用户 **joe** 的口令字段都包含一个“x”，并且实际加密口令驻留在 **/etc/shadow** 中。如果是一个非映像标准系统，用户 **root** 和用户 **joe** 会包含实际加密口令。

NIS 举例

root:3Km/o4Cyq84Xc:0:10:System Administrator:/:/sbin/sh

joe:r4hRJr4GJ4CqE:100:50:Joe User,Post 4A,12345:/home/joe:/usr/bin/ksh

+john:

-bob:

+@documentation:no-login:

-@marketing:

+:::Guest

在这个 NIS 例子中，有属于用户 **root** 和 **joe** 的特定条目，以防网络信息系统出现故障。

- 用户 **john** 在网络信息系统中的口令条目被原样并入。
- 忽略任何后来的用户 **bob** 的条目。
- 网络组 **documentation** 中任何用户的口令字段被禁用。
- 网络组 **marketing** 中的用户不通过 *getpwent*(3C) 返回，因而不允许登录。
- 其他任何人能以各自日常的口令、 Shell 和主目录进行登录，但需要有一个叫作 **Guest** 的 gecos 字段。

文件

/tcb/files/auth/*/*	将系统转换为信任系统时使用的受保护口令数据库。
/etc/passwd	由 HP-UX 使用的标准口令文件。
/etc/shadow	映像口令文件。

另请参阅

chfn(1)、 chsh(1)、 finger(1)、 login(1)、 passwd(1)、 pwck(1)、 pwconv(1M)、 useradd(1M)、 a64l(3C)、 crypt(3C)、 getpass(3C)、 getpwent(3C)、 getprpwent(3)、 authcap(4)、 shadow(4)、 limits(5)。

符合的标准

passwd：SVID2、SVID3、XPG2

名称

pcf - DDFA 软件使用的端口配置文件

说明

端口配置文件由数据通信与终端控制器设备文件访问 (DDFA) 软件使用来配置单独的终端服务器端口。该模板文件的一般名称是 **pcf** 。实际上，它为每个端口重新命名，该端口需要不同的配置值，以及调整为适合附加到该端口设备的值。端口配置文件由专用端口文件 (**dp**) 中的一个条目参考。专用端口解析程序 (**dpp**) 解析 **dp** 文件并为 **dp** 文件中的每个有效条目衍生一个出站连接守护程序 (**ocd**) 。有效的条目中的第四个字段是端口配置文件名。

主端口的配置文件是 **/usr/examples/ddfa/pcf** ，并且如果它包含的缺省值对执行端口正确则它应该只在 **dp** 文件中被引用。如果需要不同的值，则 **/usr/examples/ddfa/pcf** 应该复制到另一个目录，副本应被修改并在 **dp** 文件中被引用。建议的程序是用于创建一个目录，以便存放端口配置文件和修改过的 **dp** 文件。

请参阅 *ddfa*(7) 来了解关于如何配置 DDFA 软件的详细信息。

端口配置文件由变量名及其值组成。变量由冒号 (**:**) 结尾，但这不是强制的。一个变量和它的值可以由空格或制表符分隔。每行只允许一个变量值对。只有值可以改变。变量名不应该改变。

该文件包含以下信息：

telnet_mode: 它可以有值 **disable** 或 **enable** 。当它启用时，网络上的数据传输使用 Telnet 协议。对 DTC 而言，该选项 必须是启用。

timing_mark: 它可以有值 **disable** 或 **enable** 。当它启用时，所有用户数据传输后，发送一个 telnet 计时标记协商给终端服务器。关闭该连接前， **ocd** 等待对该计时标记协商的响应。这样确保在缓冲区清除前，所有数据都从终端服务器输出到设备。对于 DTC 它应该启用。

telnet_timer: 这样定义了以秒为单位的时间，该期间内软件等待 telnet 计时标记和二进制协商的响应。如果计时器过期，则有一条错误消息记录到日志 **/var/adm/syslog** 并且该错误被传送到用户应用程序。

binary_mode: 它可以有值 **disable** 或 **enable** 。当它启用时，网络上的数据传输是二进制模式并且对特殊字符（比如 XON/XOFF）的处理为禁用。

由于没有流控制，在 **binary_mode** 启用时无法保证数据完整性。

注意，即使 **binary_mode** 禁用，也可以在任何时间通过在 **termio** 数据结构中设置 **IXON** 为 **0** 来进行协商。

open_tries: 它定义了放弃前软件尝试打开连接的次数。如果该值为 **0** 则软件“永远”尝试（大约 68 年）。如果重试进程失败，则有一条错误消息记入日志 **/var/adm/syslog** 并且该错误被发送给用户应用程序。

可以使用 **kill -17** *pid* 发送 **SIGUSR2** 信号到 **ocd** 以中断重试进程。

注意，如果要求 **ocd** 打开到终端服务器的连接后应用程序存在， **ocd** 继续尝试打开直到 **open_tries** 和 **open_timer** 的组合超限。

open_timer: 它定义了以秒为单位的打开尝试间的时间。如果该值为 **0** ，则 **ocd** 使用最高到 32 秒的指数重试区间（即 1 2 4 8 16 32 32 32 ...）。

close_timer: 它定义了应用程序在 pty 从属上进行的关闭调用与连接实际关闭的时刻之间的时间，该时间以秒为单位。当后台打印程序同时后台处理几个文件时，例如将该值设置为 5 秒就避免了打开和关闭连接的开销。设置足够高的值可以有效地使连接长久打开。

status_request: 它可以有值 **disable** 或 **enable** 。当它启用时，软件向附加到该终端服务器的设备发送一个状态请求，进程响应如下：

LP_OK *(0x30)*	**ocd** 继续处理。
LP_NO_PAPER (*0x31*)	**ocd** 在状态计时器的限制内重试。
LP_BUSY (*0x32*)	**ocd** 在状态计时器的限制内重试。
LP_OFF_LINE (*0x34*)	**ocd** 在状态计时器的限制内重试。
LP_DATA_ERROR (*0x38*)	**ocd** 在状态计时器的限制内重试。

status_timer: 这定义了以秒为单位的时间，该时间内软件等待对该状态请求的响应。如果计时器过期，则有一条错误消息记入日志 **/var/adm/syslog** 并且该错误被发送到用户应用程序。

eight_bit: 它可以有值 **disable** 或 **enable** 。通常，pty 处理的数据字节拆分为 7 位。如果 **eight_bit** 启用，则禁用拆分。如果 **eight_bit** 禁用，则拆分启用并且第 7 位被拆分。也可以通过使用 ioctl( ) 命令更改别名的 termio 结构来实现。

tcp_nodelay: 它可以有值 **disable** 或 **enable** 。当它启用时，数据在它被接收到时发送到 LAN 。如果软件发送包比服务器接收包的速度快则它可以禁用。

缺省值包括：

telnet_mode	**enable**
timing_mark	**enable**
telnet_timer	**120**
binary_mode	**disable**
open_tries	**1500**
open_timer	**30**
close_timer	**5**
status_request	**disable**
status_timer	**30**
eight_bit	**disable**
tcp_nodelay	**enable**

警告

为了确保命令（比如 *ps* ）显示正确的设备文件名（也就是 *pseudonym* ，）所有别名应该放到目录 **/dev/telnet** 中。如果没有在此目录中指定要替换的别名，则不能保证有许多命令的设备文件名正确显示。

此外，为了确保命令（比如 **w** 、 **passwd** 、 **finger** 和 **wall** ）正确工作，每个别名前 17 个字符必须唯一（包括目录前缀 **/dev/telnet/** ）。如果别名前 17 个字符不唯一，则不能保证许多命令正确运行。

文件

/usr/sbin/dpp

/usr/sbin/ocd

/usr/sbin/ocdebug

/var/adm/dpp_login.bin

/var/adm/utmp.dfa

/usr/examples/ddfa/dp

/usr/examples/ddfa/pcf

另请参阅

dpp(1M)、ocd(1M)、ocdebug(1M)、dp(4)、ddfa(7)。

名称

pfs、PFS - 可移植文件系统

说明

可移植文件系统或 PFS 允许访问各种 CD-ROM 文件系统。当前受支持的文件系统包括: **iso9660** 、 **high sierra** 、 **RockRidge Interchange** 。

PFS 程序包由七个程序组成:

pfs_mountd 负责维护本地挂接和远程挂接。PFS 客户端和 PFS 服务器都必须运行它。**pfs_mountd** 程序验证参数，并生成 **pfs_mountd.rpc** 文件。

pfs_mountd.rpc 是与 **pfs_mountd** 相关联的 RPC 服务器代码。它应直接执行。

pfsd 响应所有对给定 CD-ROM 文件系统的客户端请求。 **pfsd** 需要运行在所有指定为 PFS 服务器的系统上。 **pfsd** 验证参数，并衍生 **pfsd.rpc** 。

pfsd.rpc 与 **pfsd** 相关联的 RPC 服务器代码。它应直接执行。

pfs_exportfs 使 PFS 客户端挂接可以使用本地目录。

pfs_mount 本地挂接或从服务器挂接 CD-ROM 文件系统。

pfs_umount 本地卸载或从服务器卸载 CD-ROM 文件系统。

客户端文件访问调用被转换为 PFS 协议请求，并通过网络发送给服务器系统。服务器接收请求，并进行实际的文件系统操作，然后将响应发送回客户端。

可移植文件系统的操作是以状态方式进行的，它使用在外部数据表示 (XDR - rfc1014) 协议顶层上构建的远程过程 (RPC - rfc1057) 调用。 RPC 协议为网络安全提供需要交换的版本参数和验证参数。

通过在服务器的 **/etc/pfs_exports** 文件中添加特定文件系统条目，并运行 *pfs_exportfs*(1M) ，服务器可以授予某些客户端访问特定文件系统的权限。

客户端可以使用 **pfs_mount** 命令获得对文件系统的访问权。客户端挂接文件系统后，服务器向客户端发送它们访问和创建的每个文件（或目录）的文件句柄。如果在服务器上卸载光盘，文件句柄就过时，而远程请求将返回过时的文件句柄消息。

服务器还可以是在网络上挂接了文件系统的客户端计算机，但是它的客户端不能访问那些文件系统。相反，客户端必须直接从它所驻留的服务器挂接文件系统。

错误

通常，将向客户端返回在服务器上检测到的物理磁盘 I/O 错误，以便进行相应的操作。如果服务器停机或不可访问，客户端将会收到以下消息:

PFS server ***host*** **not responding, retrying...**

它将重试 4 次，最后返回失败。

警告

PFS 是过时形式，任何 HP-UX 发行版都不再支持它。下一版本的 HP-UX 发行版将停止发布 PFS 接口。

PFS 由 Young Minds，Inc.（现已解散）开发，最初 HP 采用它是为了支持 iso9660 CD-ROM 文件系统上的 Rock Ridge Interchange 文件系统格式。现在， HP-UX CDFS 文件系统类型和 HP-UX 的标准文件系统命令提供了等同的功能。

PFS 具有众所周知的功能和性能问题。建议 HP 客户停止使用 PFS 接口。使用标准的 HP-UX 命令并将文件系统类型指定为 **cdfs**，可访问所有的 CD-ROM 文件系统。例如，要挂接一个 CD-ROM 文件系统，可以使用：

mount -F cdfs /dev/dsk/c0t0d4 /cdrom

无须将 cdfs 文件类型视为与其他文件系统类型不同的类型；因此，访问各种 CD-ROM 文件系统格式无须使用特殊守护程序或命令。

请参阅 *mount*(1M) 和 *mount_cdfs*(1M) 。

作者

pfs 由 Young Minds，Inc. 开发。

文件

/etc/pfs_exports

另请参阅

pfs_exports(5)、 fstab(4)、 pfs_mount(1M)、 pfs_exportfs(1M)、 pfsd(1M)。

名称

ppp.Auth - ppp 验证文件格式

说明

文件 **/etc/ppp/Auth** 包含链路级验证协议 **CHAP** （“"挑战握手验证协议"”）和 **PAP** （“"口令验证协议"”）的 HP PPP 实现所使用的值。本 CHAP 实现和 PAP 实现遵循 RFC 1334 PPP 验证协议。

CHAP 是一种更强的验证机制，无论任何时候，只要可行，都应优先使用 CHAP，而不是 PAP。

格式

每种验证规范由其最多可包含 1023 个字符的行构成。注释以“#”开头，并延伸至行末尾，空行或以“#”开头的行被忽略。字段之间用水平空白字符（空格或制表符）分隔。

如果 **pppd** 使用 CHAP 验证，那么行上的第一个单词必须与 CHAP 挑战或响应包中接收的对等端的 *Name* 相匹配，而第二个单词用作 *Secret* 。如果 **pppd** 使用 PAP 验证，行中的首单词必须与发出或收到的 PAP 验证请求包中的 **Peer-ID** 相匹配，而第二个单词用作 *Password* 。发送的 CHAP 包中的 Name 或发送的 PAP 包中的 Peer-ID 的缺省值是运行 **pppd** 的计算机中的 *hostname*(1) 。

在匹配之前，Name/Peer-ID 字符串和 Secret/Password 字符串中间的 *^x* 被转换为相应的控制字符，而 **\xxx** 表示与八进制数字 **xxx**相对应的字符。其他特殊序列如下:

\s 匹配空格字符 (ASCII 0x20)。

\t 匹配水平制表符 (ASCII 0x09)。

\n 匹配换行符 (ASCII 0x0a)。

\r 匹配回车符 (ASCII 0x0d)。

字段的含义如下:

name 发送或收到的 CHAP 挑战报文或响应消息的 Name 字段，或发送或收到的 PAP 验证请求消息的 Peer-ID 字段。对于发送的包，它是主机名，除非被 **pppd name** 选项覆盖。

secret 对等端也知道的私用密钥。

optional address restrictions

由零个或多个模式组成的模式集对地址进行限制，使我们能够将它与已命名的对等端一起使用。模式之间用空格或制表符分隔，并被从左向右解析。每个模式可以用感叹号开头，以表明禁止使用后面的模式。模式的其余部分由数字和句点构成，开头和末尾可以包含星号，表示可以与任何内容相匹配。如果任何模式都不匹配，只有当最后一个模式以感叹号开头时，地址才能使用，否则不能使用。

这种可选的地址限制功能仅适用于 IPv4 地址。

举例

下面的 **Auth** 为 **pppd** 提供密钥，对等端转换为 other-host、robin 或“Jack’s machine”时使用相应的密钥。

#

```
# Auth - PPP authentication name/secret file
# Format:
#name       secret      optional address restrictions
other-host  secret-key  !137.175.9.2 137.175.9.*/0xffffff00
robin       dK3ig8G8hs         137.175.11.4
Jack's\smachine        I\sam\sa\sjelly\sdonut.
```

安全考虑

文件 **/etc/ppp/Auth** 应为模式 600 或模式 400，而且由超级用户拥有。

作者

ppp.Auth 由 Progressive Systems 开发。

另请参阅

ppp.Devices(4)、 ppp.Dialers(4)、 ppp.Filter(4)、 ppp.Keys(4)、 ppp.Systems(4)、 services(4)、 pppd(1)、 RFC 792、 RFC 1548、 RFC 1332、 RFC 1334。

名称

ppp.Devices - PPP 物理设备描述文件格式

说明

文件 **/etc/ppp/Devices** 将拨号程序类型与物理设备和速率相关联。当呼叫相邻计算机时， **pppd** 将检查这个文件。如果没有查找到合适的速率，或者具备合适速率的所有设备都忙， **pppd** 稍后将重试。

格式

每个条目占一行；空白行被忽略。注释以“#”开头，并延伸至行末尾。大小写的区分很重要。行的各个字段用水平空白字符（空格或制表符）分隔。

每个条目包含三个或更多个字段，字段的顺序如下：

dialer 可以是字符串“Direct”，也可以是调制解调器拨号所使用的交谈脚本（可以在 **Dialers** 中找到），还可以是外部拨号程序的名称。

device 设备名称位于 **/dev** 目录（ ttya 和 **cua** 等）。SnapLink 连接的设备名称后接斜线和所使用的端口号（ rsd2a/0 和 **rrz4a/2** 等）。

speed 同步连接的波特率，或者需要与 **Systems** 中的条目的 speed 字段进行匹配的字符串（当 **Systems** 设备字段设置为 ACU 时）。速率必须是有效的异步波特率数字（与 **<sys/ttydev.h>** 中的数字相同），或者必须以这些数字（2400、38400、19200-PEP 等）开头，或者必须是 SnapLink 硬件所具备的速度（9600、56000、64000 和 1536000 等）。

optional parameters
设备的任何特殊处理。当前支持的值包括：

xonxoff 将线路设置为使用带内（“软件”）流控制，使用字符 DC3（^S、XOFF、ASCII 0x13）来停止流，并使用 DC1（^Q、XON、ASCII 0x11）恢复流。缺省时不使用流控制。对于出站连接来说，可以在 **Devices** 中或 **pppd** 命令行上进行指定。

internal-clocking SnapLink 将提供同步时钟信号。缺省情况下，它需要调制解调器、CSU/DSU 或调制解调器代用器提供时钟信号。内部时钟不能与 SnapLink 上的 RS-232 电缆一起使用。

32-bit-fcs SnapLink 将计算传输帧的 32 位 FCS 的值，并用 32 位 FCS 计算检查收到的帧。在建立连接时，这是不可协商的。只有在 SnapLink 上运行同步 PPP 时，32 位 FCS 才是可用的。

min-flags=*minflags* SnapLink 应在数据帧之间插入的附加 HDLC 标志字符的数量。缺省值和最小值都是 2，最大值是 16。

ignore-cd 忽略 CD（载波检测，又称数据载波检测 (DCD)）信号的状态。这对于不支持 CD 但希望在专用线路上运行 PPP 的系统很有用。

外部拨号程序

运行外部拨号程序需要以下参数:

device name	Device 条目的 Devices 字段的内容。
speed	Systems 和 Devices 条目中的 Speed 字段的内容。
telephone number	Systems 条目的 Phone Number 字段的内容。
optional parameters	从 Devices 条目的 Optional Parameters 部分复制而来。

如果外部拨号程序以状态 0 退出，拨号尝试被视为已经成功。其他退出状态表示失败。

举例

```
#
#       Devices - PPP devices file
#
#Dialer     device      speed       Optional parameters
T2500-PEP               cua         19200-PEP           rtscts
T1600       cub         38400       rtscts
Direct      rsd0a/0     1536000     internal-clocking
Oddball     rsd0a/1     64000       cua 9600 5551212
```

在本示例的最后一行，SnapLink 端口 1 上的 64Kb 同步调制解调器有一个附加到工作站的端口“a”的异步拨号程序接口。Systems 行的内容如下所示:

```
host Oddball rsd0a/1 64000 0
```

进行以下调用时，必须有一个名为 **/etc/ppp/Oddball** 程序（或一个可执行的 Shell 脚本）使用调制解调器进行拨号:

```
Oddball rsd0a/1 64000 0 cua 9600 5551212
```

如果 *debug* 级别设置为 2 或更高，将打印未识别的各可选参数之警告信息。

外部拨号程序是作为 **root** 调用的，因此必须为其内容和文件保护采用相应的安全措施。

作者

ppp.Devices 由 Progressive Systems 开发。

另请参阅

ppp.Auth(4)、 ppp.Dialers(4)、 ppp.Filter(4)、 ppp.Keys(4)、 ppp.Systems(4)、 pppd(1)、 RFC 1548、 RFC 1332、 RFC 1144、 RFC 1055。

名称

ppp.Dialers - PPP 拨号程序描述文件格式

说明

文件 **/etc/ppp/Dialers** 描述如何通过外发 PPP 呼叫调用与 UNIX 系统相连的各类调制解调器。当呼叫相邻的计算机时， **pppd** 会检查这个文件。

当 **pppd** 从 **Systems** 选择一行时，它使用“speed”字段来选择 **Devices** 中的条目，并使用“dialer”字段选择 **Dialers** 中的条目。然后， **pppd** 解释拨号程序描述中的“chat script”字段。

格式

每个条目占一行；空行忽略。注释以“**#**”开头，并延伸至行末尾。为了进行匹配，“dialer”字段区分大小写，“chat script”字段中的字符串也是如此。行上的各个字段之间用水平空白字符（空格或制表符）分隔。交谈脚本以反斜杠（“\”）结束，下一行被视为交谈脚本的延续。只有在交谈脚本之中才能延续。

每个条目都必须包含以下字段，其顺序如下所示：

dialer 拨号程序的名称，需要与 **Devices** 中的 dialer 字段进行匹配。

chat-script **pppd** 保存与调制解调器之间的会话描述。

交谈脚本特殊情况

交谈脚本是用空格分隔的 expect-send 对列表。每个对（至少）包含一个字段，期望“remote”终端发送请求，并发送一个字段以作出响应。除非“send”字符串以 **\c** 结尾， **pppd** 将通过发送一个回车符 (ASCII 0x0d) 紧随其后。

交谈脚本的形式为“expect send expect send ...”或“expect-send-expect send ...”。在这里，如果连字符前面的 expect 不能与接收的文本相匹配，将执行连字符后面的 send。

交谈脚本中可以使用一些特定单词，以在 **pppd** 拨号时控制其行为。ABORT 和 TIMEOUT 必须在交谈脚本的“expect”阶段。

ABORT *abort-string* 如果 **pppd** 在执行交谈脚本的剩余部分时发现 *abort-string*，将终止拨号，并在日志文件写入失败信息。

TIMEOUT *timeout-time* 在执行当前交谈脚本时，等待 *timeout-time* 秒，以期待获得响应，之后才认为拨号连接超时。脚本的超时固定为 60 秒。

" P_*WORD* 的 expect-send 对相应地设置行的奇偶校验：

P_AUTO 根据“expect”字符串中接收到的字符中的奇偶校验设置传输奇偶校验。这是缺省值。

P_ZERO 传输奇偶校验位设置为零的字符（8 位，不带奇偶校验位）。

P_ONE 传输奇偶校验位设置为 1 的字符。

P_EVEN 传输带有偶校验的字符。

P_ODD 传输带有奇校验的字符。

在“expect”字符串或“send”字符串的中间，ˆ*x* 被转换为相应的控制字符，且 *****x* 被转换为 *x* 。其他一些特殊序列为:

\s 发送或接收空格字符 (ASCII 0x20)。

\t 发送或接收水平制表符 (ASCII 0x09)。

\n 发送或接收换行符 (ASCII 0x0a)。

\r 发送或接收回车符 (ASCII 0x0d)。

**** 发送或接收反斜杠字符 (ASCII 0x5c)。

\ˆ 发送或接收克拉符 (ASCII 0x5e)。

ˆ*character* 发送或接收 Ctrl 字符（ASCII 0x00 到 0x1f）。

*****ddd* 发送或接收用八进制位表示的字符。

\p 在进行之前暂停 0.25 秒（仅用于发送）。

\d 进行之前暂停两秒（仅用于发送）。

\K 发送中断（0.25 秒零比特发送）。

\M 禁止挂起（设置 CLOCAL 或 LNOHANG）。

\m 启用挂起（解除 CLOCAL 或 LNOHANG）（缺省情况）。

\c 发送前导字符串后，不追加回车符（仅用于发送）。

\q 不在调试输出或日志输出中打印发送成功的字符串（如口令等）。后继的 **\q** 序列切换“无提示”模式。

\T 插入（在 **Systems** 的第五字段中找到）电话号码。

举例

```
#
#       Dialers - PPP dialers file
#
#Dialer Chat script
T1600       ABORT NO\sCARRIER ABORT NO\sDIALTONE ABORT BUSY \
            ABORT RRING\r\n\r\nRRING\r\n\r\nRRING \
            ABORT ERROR TIMEOUT 5 "" AT OK-AT-OK \
            ATS111=0DT\T TIMEOUT 30 CONNECT
#
T2500-PEP \
            ABORT NO\sCARRIER ABORT NO\sDIALTONE ABORT BUSY \
```

```
            ABORT RRING\r\n\r\nRRING\r\n\r\nRRING \
            ABORT ERROR TIMEOUT 5 "" AT OK-AT-OK \
            ATS111=0DT\T TIMEOUT 30 CONNECT\sFAST
    #
    USRv32bis \
            ABORT ERROR ABORT NO\sANSWER ABORT NO\sCARRIER \
            ABORT BUSY ABORT RRING\r\n\r\nRRING\r\n\r\nRRING \
            ABORT NO\sDIAL\sTONE TIMEOUT 5 "" AT&F \
            OK-ATQ0-OK ATB0E0X7&B1&H1&I0&K3&R2&S1 OK-AT-OK \
            ATS01=1S02=255S19=0 OK-AT-OK ATDT\T TIMEOUT 30 \
            CONNECT
```

作者

ppp.Dialers 由 Progressive Systems 开发。

另请参阅

ppp.Auth(4)、 ppp.Devices(4)、 ppp.Filter(4)、 ppp.Keys(4)、 ppp.Systems(4)、 pppd(1)、 RFC 1548、 RFC 1332、 RFC 1144、 RFC 1055。

名称

ppp.Filter - PPP 包过滤器规范文件的格式

说明

文件 **/etc/ppp/Filter** 描述请求 PPP 链路的管理方式。缺省情况下，任何类型的包都将启动链路（如果已断开，则重新连接至远程终端）；任何包都可以遍历链路，任何数据包都可以重置空闲计时器，计时到期时可能会导致链接关闭。这种设置并非总是合适的，因此过滤器文件允许根据包的类型和包的来源或目的地来进行单独控制。对于链接运行的三个阶段中的任一阶段，都可以指定这些选择条件：启动链路，在链路上传送包和由于不活动而关闭链路。还可以使用同一条件来设置包日志记录详细程度。

格式

注释行以“#”开头并一直延伸到行的结束。空行或以“#”开头的行都被忽略。主机名规范忽略大小写，但其他任何地方都需要区分大小写。字段之间用水平空白或垂直空白（空格、制表符或换行符等）分隔。

如果行的开头是主机名、IPv4/IPv6 地址、特殊字“default”或 IPv6 的特殊字“default6”，则该行将被视为一组新过滤规范的开始。过滤规范将应用于连接此主机与对等端之间的点对点链路上的任何包，对等端以初始主机名或 IPv4/IPv6 地址命名。过滤器文件第一列中的主机名或 IPv4/IPv6 地址表示点对点（PPP 或 SLIP）链路的远程终端上的对等点（系统、路由器或终端服务器）。过滤器文件第一列中的主机名或 IPv4/IPv6 地址（它们与链路对等端相关联）与链路上传输的包的 IPv4/IPv6 源地址和目标地址没有关联。如果链路对等端的地址与过滤器文件中第一列指定的名称或地址不匹配，那么将使用 IPv4 特殊字“default”或 IPv6 特殊字“default6”后的过滤规范。

如果一个新行后面是空白，那么这个行仍是现行过滤规范的延续。

可以使用四个关键字来描述 **pppd** 对特定包作出的响应动作。

bringup 描述将引发调用连接和初始化连接的包。还必须将这种数据包限定为跨链接“传递”，要么明确指定，要么包含在“传递”部分的较大类中。

pass 描述允许在已建立的连接上遍历链路的包。只有被传递的包才能引发启动链路。 未被传递的包是有选择地记录的，然后被忽略。

keepup 描述将重置空闲计时器的包，并保持线路连通。

log 描述包头或内容将被记录到日志文件的包。

每个操作关键字后跟节，之间用空白字符分隔，用于描述符合该操作条件的数据包。将按照在文件中的显示顺序对每一节进行处理，每一节都包含针对所遇到的数据包的限制条件或权限。发现与此处讨论的包相匹配的模式或条件后， **pppd** 将执行指定的动作，并忽略列出的其他参数节（即包含或计算快捷方式）。

参数节可能包含 IP 协议编号，还可以包含用连字符分隔的 TCP 或 UDP 端口号范围连同 **/tcp** 或 **/udp** 限定符、表示 ICMP/ICMPv6 报文类型或编码的编号（可以在 **<netinet/ip_icmp.h>** 和 **<netinet/icmp6.h>** 中找到）连同“/icmp”或 对于 ICMPv6 来说的“/icmp6”限定符、与 **/etc/services** 中的条目相对应的服务名、主机或网络的名称或 IP 地址或特殊字“all”，除了“log”动作缺省的特殊字是“!all”外，其他动作缺省的特殊字都是“all”（通常情况下，无须使用“all”，为了方便起见，如果最后一个参数节不是无效的， **pppd** 自动在参数节列表后添加一个“!all”，如果最后一个参数节是无效的，将在参数节列表后添加一个“all”。例如，对于典型的“日志记

录” 操作，这会明显导致 只有与列出的节相匹配的数据包才被记录到日志中，而其他数据包则不会。对于典型的“传送”操作，将导致某些包受到限制，而其他所有的包都可以通过）。

对于 IPv4 包过滤，如果网络是用名称或地址指定的，那么必须能够指定相应的网络掩码，即使掩码的大小与此类网络缺省的掩码的大小不相同。网络掩码与节内其他“and”条件之间用斜线 (“/”) 分隔，网络掩码可以指定为一系列用句点分隔的十进制数字或一个 32 位的十六进制数字。对于 IPv6 网络，应指定一个前缀或一个具有前缀长度的 IPv6 地址，用斜线 (“/”) 加以分隔。在参数节之前添加前缀感叹号 (“!”) 可以使参数节无效。

在“log”过滤器规范中，特定关键字“trace”将引发指定类型包的内容（和包头）被写入日志文件。同样在“log”过滤器规范中，特定标志“rejected”表示只有被“pass”过滤器拒绝的包才被记入日志。

由于 TCP 数据流是在初始化器向既定的接收方发送包时处于开启状态，因此 **pppd** 可以使用 telnet 或 FTP 等 TCP 应用程序来区分出站（从主机发出）和入站（从链路的另一端发来）特殊关键字“syn”允许启动对这些连接的过滤或日志记录。限定符“ecv”或“send”表明只有按照指定的方向初始化会话，才能启动和记录会话日志。特殊关键字“fin”允许过滤或记录将关闭 TCP 连接的包。

关键字“src”和“dst”用于区分端口、地址或主机名，它们分别应用于包的来源和目的地。如果两者都应用于同一参数节（例如 **.../src/dst** ），那么源与目的地的地址和/或端口都必须相匹配。

unreach= 关键字还可让未达 ICMP 目的地的报文（RFC 792 和 RFC 1122 的 3.2.2.1 节）发送至 IPv6 包的源地址，该报文具有指定的代码域，代码域可能是以下域:

net	无法访问目标网络。
host	无法访问目标主机。
prot	不支持指定的传输协议。
protocol	不支持指定的传输协议。
port	指定的传输协议（如 UDP）不能多路分离数据报，而且没有协议机制来通知发送方。
needfrag	分片是需要的，但设置为“不分片”标志。
srcfail	源路由失败。
net-unknown	目标网络未知。
host-unknown	目标主机未知。
host-isolated	源主机被孤立。
net-prohibited	在管理上禁止与目标网络的通信。
host-prohibited	在管理上了禁止与目标主机的通信。
net-tos	指定类型的服务无法访问目标网络。

host-tos 指定类型的服务无法访问目标主机。

unreach= 关关键字还可让未达 ICMPv6 目的地的报文（RFC 2463 的 3.1节）发送至 IPv6 包的源地址，该报文具有指定的代码域，代码域可能是以下域：

noroute6 没有能够到达目的地的路径。

admin-prohibited 在管理上禁止了与目标的通信。

addr6 无法访问目标地址。

port6 无法访问目标端口。

根据包是否含有各个 IP 选项（RFC 1122 的 3.2.1.8 节和 RFC 791 的 3.1节 (pps 16ff) ），可以使用 **ip-opt=** 关键字来选择包：

rr 记录路由用于跟踪 Internet 数据报的路径。

ts 时间戳。

security 使用安全机制来实现安全、区域化、用户组 (TCC) 和 DOD 规则兼容的处理限制码。

lsrr 使用宽松源路由根据源提供的信息路由 Internet 数据报。

ssrr 使用严格源路由根据源提供的信息路由 Internet 数据报

srcrt 使用宽松源路由或严格源路由。

any 任何 IP 选项，甚至可以是 No Operation 选项。

举例

缺省行为

下面的 **Filter** 文件描述 **pppd** 的缺省行为，没有过滤器规范文件或者文件为空时，pppd 也具有这些缺省行为：

```
#   Filter - PPP configuration file,
#   binding packet types to actions.
#   Describes the default behavior of the daemon:
default   bringup all pass all keepup all log !all

default6  bringup all pass all keepup all log !all
```

缺省行为对包不加限制，也不记录包。

Internet 防火墙

在机构网络与更大的 Internet 之间，应添加一个“pass”行来作为安全防火墙：

```
internet-gateway
    bringup !ntp !3/icmp !5/icmp !11/icmp !who !route
        !nntp !89
```

```
pass    nntp/137.39.1.2 !nntp
        telnet/syn/recv/137.175.0.0
        !telnet/syn/recv !ftp/syn/recv
        !login/syn/recv !shell/syn/recv !who
        !sunrpc !chargen !tftp !supdup/syn/recv
        !exec !syslog !route !6000/tcp/syn/send
keepup  !send !ntp !3/icmp !5/icmp !11/icmp
        !who !route !89
log     rejected
```

这个“pass”规范允许只有一个对等端而没有另一个对等端的 NNTP（Usenet 新闻）事务。它仅仅允许一个网络上的主机发来的 Telnet 会话，而拒绝其他所有到来的 Telnet、SUPDUP 和 FTP 会话，但允许所有发出的 Telnet、SUPDUP 和 FTP 会话。

它允许在本地窗口服务器上显示运行在其他地方的 X 窗口系统客户端，但禁止本地 X 客户端使用位于其他地方的显示。它拒绝所有 SUN RPC 流量，从而防止本地 YP/NIS 和 NFS 服务器受到外部侦测和文件系统挂接的影响。此外，它还拒绝本地计算机挂接驻留于其他地方 NFS 服务器的文件系统，但是这可能没有帮助，因为 NFS 使用 RFC (RFC 是一个 UDP 服务)，因此它没有 SYN 和 FIN 包，而这两个包可用来描述 TCP 流的初始化方向。它阻塞具有恶意目的流量，如果没有“!all”结尾，则意味着除显式地阻塞的流量外，其他流量都可以通过。

“bringup”和“keepup”适用于间歇拨号连接，这样各种出错状态不会导致建立链路，也不会在超过效用的情况下保持调用。OSPF（最短路径优先）路由包（RFC-1340 中的 IP 协议编号 89）将在链路上传输，但是它不会导致链路启动，也不会在链路空闲时保持链路。 Usenet 新闻流不会启动链路，但是一旦启动链路后，在一组新闻传输过程中链路将不会关掉。“log rejected”行将为每个被“pass”行阻塞的包保存一条记录，因此未能成功穿透防火墙的企图将被记录。

IPv6 过滤器应添加以下行：

```
<IPv6 link local gateway address>  #like fe80::2222
    # which type of traffic should/shouldn't bring up the line

    bringup !ntp !128/icmp6 !137/icmp6 !who !route !nntp

    # which type of packets should be passed/rejected

    pass    !nntp
            !telnet/syn/recv
    # Don't allow any packets from network whose prefix matches
    # prefix cafe.

            !cafe::1234/16
            !ftp/syn/recv !login/syn/recv !shell/syn/recv
```

```
# which type of packets should/shouldn't restart the idle timer

keepup  !send !ntp !137/icmp !who !route

# which type of packets should/shouldn't be logged

log     rejected
```

一个极其复杂的例子

下面的 **Filter** 文件通知守护程序，为了响应一个包，由 NTP、 ICMP 目的地不可达和 **rwhod** 生成的包除外，应启动与相邻对等端的连接，骨干主机除外。如果链路上只有这些类型的包，那么所有的包都可以在已启动的链路上传送。发出的包不设置空闲计时器，从对等端接收到的包将设置空闲计时器。如果因为对等端计算机故障或调制解调器故障导致不能挂断电话，（而且指定了 *idle* 命令行参数） **pppd** 将挂断连接并重试。

对于骨干主机（可能是网络接入服务提供商的服务器）这种特殊情况，只有 telnet 和 FTP 会话、SMTP 电子邮件、 NNTP 网络新闻和域名系统查询才能够启动链路或保持处于空闲状态的链路。

启动链路后，以上所有的会话和 NTP 时钟报时装置以及 ICMP 报文都可以在链路上传输。发送给特定主机的包或特定主机发出的包不能在链路上传输，除了域名系统查询和响应 B 类网络 137.75 的 42 子网上的主机的包以外，其他任何包都不能在链路上传输，也不能引发链路初始化。仅允许出站方向初始化的 telnet 和 FTP 会话。

记录各种 ICMP 出错报文（不可达，超时等）的单行描述以及报告 IP 头错误的 ICMP 报文的完整内容。记录所有 telnet 和 FTP 会话，包括包的入站尝试（尽管它们将失败，因为它们被前面的“pass”规范所排除）。也记录链路上发送给特定主机或由特定主机发送的电子邮件报文的首个包的头。

```
#
#       Filter -  PPP configuration file binding packet
#               types to actions.
#
#       For packets that would pass, these services
#       will bring up the link:
#
backbone bringup smtp nntp domain telnet ftp
#
#       Once brought up, these will pass (or not):
#
        pass    !131.119.250.104
                domain/137.175.42.0/255.255.255.0
                !137.175.42.0/0xffffff00
#                 (alternative ways of
#                       expressing subnet mask)
                !telnet/syn/recv !ftp/syn/recv
```

```
            domain smtp nntp ntp icmp telnet ftp
#
#    Packets received for the services shown will
#    reset the idle timer.
#
    keepup  !send smtp nntp domain telnet ftp
#
#    Only these messages will have headers or contents
#    logged, unless higher-level debugging is set:
#
    log    3/icmp 11/icmp 12/icmp/trace
            telnet/syn ftp/syn
            smtp/syn/terminus.netsys.com
#
default bringup        !ntp !3/icmp !who
    keepup        !send !ntp !3/icmp !who
```

建议

简单的过滤器规范可以使 **pppd** 启动和运行更快，减小包处理开销，但是包处理开销问题通常只会出现在超高速链路（如 T1）上。前面的"骨干主机"例子是为了实例示范说明，因此进行了大量的扩充，需要花费大量的时间才能将它们设置正确。可以从一个简单的过滤器规范开始，然后在需要添加特殊的过滤器规范，通常需要在观察包日志记录后才决定是否需要添加新规范。应仔细测试，确保所作的修改取得预期效果。

应注意观察包头记录，更应注意观察包内容的跟踪。选择条件必须十分准确和严格，否则日志文件将在短时间内急剧增大。此外，如果守护程序运行在无盘工作站上，或者日志文件位于挂接了 NFS 的文件系统上，过量的日志记录信息将会极大地降低 pppd 在包的高传输速率上的处理能力。请记住，NFS 写入操作是同步的。

如果指定了主机名，请确保能够在本地获得主机地址，即使连接断开。如果发现必须启动连接来解析一个域名，请考虑使用在 **Filter** 和 **Systems** 中的主机 IP 地址（用句点分隔的十进制字数字）。

如果想要指定所有域名系统流量，请使用将被扩展给 **53/tcp** 和 **53/udp** 条目的"domain"（有些 DNS 流量使用 TCP 和 UDP 两种传输方式）。使用 **!domain/tcp** 允许域名查询，但禁止域名转向。类似地，在一些厂商发行的系统上，旧版本的 **/etc/services** 将 NTP 列为 TCP 服务。在系统上安装现行的 UDP NTP 实现后，管理员可能同时保留了旧版的 **123/tcp** 条目和现行的 **123/udp** 。正确的做法是移除 **/etc/services** 中的 **123/tcp** 条目。还应在 **Filter** 中指定 **123/udp** 。

DEC ULTRIX 4.2 和其他一些系统的 **/etc/services** 文件可能没有 FTP 数据套接字条目。如果您需要记录大批数据连接和控制连接，那么应在 **/etc/services** 中添加"ftp-data"条目，或者在 **Filter** 中显式地使用 **20/tcp** 。应首选前一个方法，因为它将使日志文件条目包含"ftp-data"符号名称，而不是包含套接字/协议形式。

如果 **/etc/services** 文件缺少您认为有用的一些应用程序级别的协议，则可以在文件中插入 Assigned Numbers RFC（编号为 1340）中的条目。例如，您可以发现添加下列行十分有用。

gopher 70/tcp
gopher 70/udp
kerberos 88/tcp
kerberos 88/udp
snmp 161/tcp
snmp 161/udp
nextstep 178/tcp
nextstep 178/udp
prospero 191/tcp
prospero 191/udp
x11 6000/tcp

如果需要使用这些应用而且它们不在系统厂商发布的 **/etc/services** 文件之中。如果您需要用这种方法来扩展您的 **/etc/services** ，可以使用下面的条目。

pass !6000/tcp/syn/send

Filter 可以使用下面的条目:

pass !x11/syn/send

它具有更强的可读性。可以在 **Examples/services.ex** 中获得一个从 Assigned Numbers RFC 中挑选出来的服务编号和名称的列表。

作者

ppp.Filter 由 HP 开发。

另请参阅

ppp.Auth(4)、 ppp.Devices(4)、 ppp.Dialers(4)、 ppp.Keys(4)、 ppp.Systems(4)、 services(4)、 pppd(1)、 RFC 791、 RFC 792、 RFC 1055、 RFC 1548、 RFC 1332、 RFC 1122、 RFC 1144、 RFC 1340。

名称

ppp.Keys - PPP 加密密钥文件格式

限制

在美国进口软件中无法进行加密。 HP 的 **pppd** 命令不支持 **gw-crypt** 选项，客户可联系 sales@progressive-systems.com 获得加密功能。

说明

在 **pppd** 命令行的 **gw-crypt** 选项中命名的密钥文件，包含由 HP PPP 的链路级加密所使用的密钥值。传输前，使用 DES 和在一个密钥文件行中指定的密钥，对源地址和目标地址与该密钥文件行上的端点相匹配的数据包进行加密。接收时，使用 DES 和在一个密钥文件行中指定的密钥，对源地址和目标地址与该密钥文件行上的地址相匹配的数据包进行解密。

格式

每一个密钥说明都位于其自己的单独一行中，且该行最多有 1023 个字符。密钥文件中的注释以“#”开头，并延伸到行的结尾；空白行或以“#”开头的行将被忽略。各字段由水平空白字符（空白字符或制表符）分隔。

密钥行上的前两个单词与每一个传输的数据包和接收的数据包的源地址和目标地址进行比较。端点地址说明可能包含主机或网络名，或者是主机或网络地址。如果按名称或地址指定了一个网络，则当网络掩码的大小与该类网络的缺省大小不同时，必须指定相应的网络掩码。掩码与网络名或地址之间用一个斜线（“/”）分隔，而且掩码可指定为一系列由句点分隔的十进制数字，也可指定为单个 32 位的十六进制数字，且前面可选择性地带有一个 C 语言形式的“0x”前缀。

密钥行的余下部分是一个 56 位（14 位）十六进制数字（没有 C 语言形式的“0x”前缀），可用作指定主机或网络对之间的 DES 密钥。为了可读性，这些数字可由水平空格分隔。如果该密钥包含少于或多于 14 个十六进制数字，则该行将被忽略。如果密钥不够安全，则在日志文件中将输出一个警告消息，并仍使用该指定的密钥进行加密。

举例

下列密钥文件提供了 **pppd** 以及当对指出的主机或网络对间的通信流进行加密或解密时所使用的密钥：

```
#
# Keys - PPP encryption keys file
#
# Format:
#endpoint            endpoint                  key
frobozz.foo.com      glitznorf.baz.edu         feed face f00d aa
147.225.0.0          38.145.211.0/0xffffffc0   b1ff a c001 d00d 1
128.49.16.0/0xffffff00   198.137.240.100       0123456789abcd
193.124.250.136      143.231.1.0/0xffffff00    e1c3870e1c3870
```

建议

避免使用脆弱或半脆弱的密钥。这些是脆弱的 DES 密钥：

00000000000000
FFFFFFFFFFFFFF
1E3C78F1E3C78F
E1C3870E1C3870

这些是半脆弱的 DES 密钥:

01FC07F01FC07F
FE03F80FE03F80
1FC07F00FE03F8
E03F80FF01FC07
01C007001E0078
E003800F003C00
1FFC7FF0FFC3FF
FE3FF8FFE1FF87
003C00F001C007
1E007800E00380
E1FF87FF1FFC7F
FFC3FF0FFE3FF8

安全考虑

密钥文件应为模式 600 或 400,并且应由根用户所有。

尽管数据包的 TCP、UDP 或 ICMP 头部分与用户数据部分一同加密,但数据包的 IP 头部分不加密。这就允许加密的数据包遍历正常的 Internet 网络,但它允许探听器使用自己的端点分析通信流。

由于 TCP、UDP 或 ICMP 头部分被加密,因此数据包路径中基于协议的过滤器将无法辨别它是 SMTP 还是 Telnet 或任何其他网络服务。这就意味着,如果防火墙允许端点间的所有通信流通过,而不管通信流类型,则加密的通信流将仅能透过数据包过滤防火墙。当将 HP-UX 系统的 PPP/SLIP 软件配置为加密通信流的端点网关时,它们在应用自己已配置的数据包过滤规则之前,对进入的加密通信流进行解密。

作者

ppp.Keys 由 Progressive Systems 开发。

另请参阅

ppp.Auth(4)、 ppp.Devices(4)、 ppp.Dialers(4)、 ppp.Filter(4)、 ppp.Systems(4)、 pppd(1)、 RFC 792、 RFC 1548、 RFC 1332、 RFC 1334。

名称

pppoec.conf - PPPoE（以太网上的点对点协议）客户端配置文件

说明

pppoec.conf 是 **pppoec** 守护程序的配置文件。该文件由 **pppoec** 读取来初始化客户端。缺省文件为 **/etc/ppp/pppoec.conf** 。注意，没有该配置文件不能运行 **pppoec** 。 **pppoec.conf** 文件中的每个条目由一个新行分隔。空行及以 **#** 开头的行被忽略。

参数

pppoec.conf 包含下列参数：

service 指定该客户端需要的服务的名称。 **any** 的值暗含了准备接收来自接入集线器的客户端。

acname 指定希望有利于服务的接入集线器的名称。

host-unique 指定该客户端使用的标记来将接入集线器的响应（PADO 或 PADS）唯一关联到特定的客户端请求（PADI 或 PADR）。 **pppoec** 生成一个唯一的二进制值并将该值包含到 PADI 或 PADR 包中的 **Host-Uniq** 标记。

retry-number 当接入集线器没有响应该客户端的请求时，指定 **pppoec** 转发 PADI 或 PADR 包的尝试次数。缺省值为 3。

timeout 指定以秒为单位的超时值。在重新发送 PADI/PADR 包之前， **pppoec** 等待该期间来接收 PADO 或 PADS 包。缺省值为 15 秒。

pppd-options 指定 **pppd** 的命令行选项。有关详细信息，请参阅 *pppd*(1) 。 pppd 的 **mru** 选项将被包含并且它将被设置为 **1492** 。

pppoec.conf 文件示例

```
[ lan4 ]
service=any
acname=gatt3
#host_unique=0
timeout=1200
retry-number=4
pppd-options=mru 1492 debug 11

[ lan0 ]
service=any
acname=gatt3
#host_unique=0
timeout=1200
retry-number=4
```

```
pppd-options=mru 1492 debug 11
#enable_ipv6=1
```

作者

pppoec.conf 由 HP 开发。

文件

pppoesd	PPPoE 服务器守护程序
pppoerd	PPPoE 中继
pppoec	PPPoE 客户端
pppd	PPP 守护程序

另请参阅

pppd(1)、 pppoec(1)、 pppoerd(1M)、 pppoesd(1M)。

名称

pppoerd.conf - PPPoE（以太网上的点对点协议）中继配置文件

说明

pppoerd.conf 是 **pppoerd** 守护程序的配置文件。通过 **pppoerd** 读取文件以初始化中继。缺省值为 **/etc/ppp/pppoerd.conf** 。注意，不能在没有此配置文件的情况下运行 **pppoerd** 。 **pppoerd.conf** 文件中的每个条目通过换行符分隔。将忽略空白行和以 **#** 开头的行。

参数

pppoerd.conf 包含下列参数：

server-intf	指定服务器接口的名称；如： **lan0** 。
client-intf-list	指定一个或多个客户端接口名称的列表。

pppoerd.conf 文件示例

```
server-intf=lan4
client-intf-list=lan2 lan0
```

作者

pppoerd.conf 由 HP 开发。

文件

pppoesd	PPPoE 服务器守护程序
pppoerd	PPPoE 中继
pppoec	PPPoE 客户端
pppd	PPP 守护程序

另请参阅

pppd(1)、 pppoec(1)、 pppoerd(1M)、 pppoesd(1M)。

名称

pppoesd.conf - PPPoE（以太网上的点对点协议）服务器配置文件

说明

pppoesd.conf 是关于 **pppoesd** 守护程序的配置文件。该文件由 **pppoesd** 读取，用于初始化服务器。缺省文件为 **/etc/ppp/pppoesd.conf** 。用户可利用这个文件，为 **pppoesd** 在 PPPoE 会话期间使用的每个网络接口设置参数。**pppoesd.conf** 文件中的每个条目具有以下格式：

parameter=*value*

配置文件中的每个条目以一个空白行分隔。程序将忽略空白行与 **#** 开头的行。

参数

pppoesd.conf 包含下列参数：

service 指定服务器可提供给客户端的服务的名称。值 **any** 表示服务器准备向客户端提供任何服务。

acname 指定用户希望从中获得可用服务的接入集线器的名称。

host-unique 指定一个客户端标记，用于唯一关联一个接入集线器（PADO 或 PADS）对于一个特定客户端请求（PADI 或 PADR）的响应。

retry-number 指定当接入集线器不响应客户端请求时， **pppoec** 尝试转发一个 PADI 或者 PADR 数据包的次数。

timeout 指定超时的秒数。在转发一个 PADI（或 PADR）数据包之前， **pppoec** 等待接收一个 PADI 或者 PADR 数据包的持续时间。缺省值为 15 秒。

pppd-options 指定 **pppd** 的命令行选项。有关详细信息，请参阅 *pppd*(1) 。

pppoesd.conf 文件示例

```
[ lan4 ]
service=any
acname=gatt3
ac_cookie=1
#host_unique=0
timeout=1200
#retry-number=3
local-ipv4-address=1.2.3.4
ipv4-address-pool=5.6.7.8 - 5.6.7.101
pppd-options=mru 1492 debug 11
# Following entries are related to PPPoEv6
local-ipv6-identifier=::9
ipv6-identifier-pool=::10 - ::100
```

作者

pppoesd.conf 由 HP 开发。

文件

pppoesd	PPPoE 服务器守护程序
pppoerd	PPPoE 中继
pppoec	PPPoE 客户端
pppd	PPP 守护程序

另请参阅

pppd(1)、 pppoec(1)、 pppoerd(1M)、 pppoesd(1M)。

名称

ppp.Systems - PPP 相邻系统说明文件格式

说明

/etc/ppp/Systems 文件说明如何使用 PPP 与相邻系统连接。

格式

条目为一行；忽略空白行。以“#”开头的行，直至结尾均为注释。指定主机名时忽略大小写的区别，但在其他位置为区分大小写。行中的字段被水平空白字符分隔（空格或制表符）。如果交谈脚本以反斜杠（“\”）结束，下一行被认定为交谈脚本的延续。只有在交谈脚本之中才能延续。

每个条目必须按如下顺序包含六个字段：

name 目标机器的主机名或 IP 地址，可以被本地解析。

when 字符串，指示系统可以被调用的一周中的天数和一天中的时间（例如，MoTuTh0800-1740）。天数部分可以为包含 Su、Mo、Tu、We、Th、Fr 或 Sa 的列表。天数也可以为 Wk 指示周中某天（与 MoTuWeThFr 相同）或 Any（与 SuMoTuWeThFrSa 相同）。

可以指定一定时间范围（例如，0800-1230)。如果未指定时间，则允许任何时间的调用。

注意允许跨越 0000 时间范围。例如， 0800-0600 表示允许除上午 6 点到 8 点间之外的任何时间。

允许使用由竖线 (|) 分隔的多重时间指定。例如，Any0100-0600|Sa|Su 表示系统可以在每天的上午 1 点到 6 点之间、或星期六和星期日的任意时间调用。

天数和时间条目（序列）后可以跟随分号，最多为使用连字符分隔的三位数字：

one 如果仅有一个数字后面跟随分号，则当作再次调用的延时，即在调用失败后重试的起始时间（以秒为单位）。例如，Any;60 表示可以在任意时间调用，但在出现失败后尝试再次调用之前等待 60 秒。如果重新调用失败， **pppd** 将重试前的延时加倍。如果没有指定初始重试延时，假定为 10 秒。

two 如果分号后有两个数字，第二个数字用作最大再次调用延时，即再次尝试调用的最大延时（以秒为单位）。每次失败调用后，重试延时将加倍直至达到此值，此后等待该最大的秒数然后再次调用。如果没有指定最大重试延迟，则假定为 3600 秒。

three 如果分号后有三个数字，第一个用作回调延时，第二个用作再次调用延时，第三个用作最大再次调用延时。回调延时是尝试重新建立先前活动连接的等待的时间（以秒为单位），此连接是被突然的行断开结束（日志文件中为 Hangup 或 SIGHUP 事件）。缺省值为回调之前没有延时。

在调用失败之后的延时期间，任何写入到 **pppd.log** 的 7 级的调试消息后将追加“dial failed”消息。

device 如果设置为“ACU”，将可能使用 **Devices** 中的任何具有匹配速率的设备。设备的拨号程序交谈脚本将首先执行，后面为 **Systems** 交谈脚本。

如果设置为 **/dev** 目录中的设备名称（ tty00 、 **cua** 等等），则 **Devices** 中的 **Direct** 相应条目可能是一个选项，不考虑 **Dialers** ，仅执行 **Systems** 交谈脚本。

如果设置为“tcp”，则其后必须为斜线，然后是作为 PPP 链接目标的系统的主机名或 IP 地址，接下来为另一个斜线，最后是联系远程 PPP 守护程序的套接字数。

speed 连接速率。如果设备字段为 ACU，速率字段将是匹配 **Devices** 中条目的字符串。速率必须为有效速率数或以之开头（2400、38400、19200-PEP 等等）。如果设备字段为“tcp...”或“telnet...”，忽略速率字段，但是必须作为占位符存在。

phone number

在拨号程序脚本中取代 **\T** 转义序列的值。如果设备字段命名 **/dev** 中的条目，则电话号码字段可选。如果设备字段为“tcp...”或“telnet...”，存在电话号码时，该字段也忽略，但必须作为占位符存在。

chat script 对由远程机器保持的 **pppd** 会话的说明。

特定交谈脚本

交谈脚本期望远程终端以单词的形式发送，然后发送回应单词。除非是以 **\c** 结尾的“send”字符串， **pppd** 后将是回车字符 (ASCII 0x0d)。

交谈脚本为“expect send expect send ...”或“expect-send-expect send ...”，连字符之后的“send”在前置“expect”与接收文本匹配失败后执行。

这类特殊单词可在交谈脚本“send”字符串中使用，用于在 **pppd** 试图拨号时控制行为。ABORT 和 TIMEOUT 必须位于交谈脚本的“expect”阶段。

ABORT *abort-string* 如果 **pppd** 在执行交谈脚本的剩余部分时看到 *"abort-string"*，则退出拨号尝试并在日志文件中记录失败。

TIMEOUT *timeout-time* 执行当前交谈脚本时，失败后重新尝试拨号前，等待 *"timeout-time"* 秒数以期望回应。写入有固定的 60 秒超时。

' **"'** **P_***WORD* 的预期发送对相应设置此行的奇偶:

P_AUTO 根据接收到的“expect”字符串中字符的奇偶校验设置传送奇偶校验。这是缺省值。

P_ZERO 奇偶校验位设置为零的传送字符（8 位，无奇偶）。

P_ONE 奇偶校验位设置为 1 的传送字符。

P_EVEN 偶数奇偶校验的传送字符。

P_ODD 奇数奇偶校验的传送字符。

反引号字符 (**‘**) 括起的程序名称在继续之前执行。如果程序在交谈脚本对的“send”阶段运行，其标准输出将在程序退出时发送到对等端。程序退出时继续交谈脚本的处理。

在“expect”字符串和“send”字符串中， **^***x* 转换为合适的控制字符， *****x* 转换为 *x* 。其他特殊序列为:

\s 发送或接收空格字符 (ASCII 0x20)。

\t 发送或接收水平制表位字符 (ASCII 0x09)。

\n 发送或接收换行字符 (ASCII 0x0a)。

\r 发送或接收回车字符 (ASCII 0x0d)。

**** 发送或接收反斜杠字符 (ASCII 0x5c)。

\^ 发送或接收 carat 字符 (ASCII 0x5e)。

^*character* 发送或接收单字符 Ctrl-字符（ASCII 0x00 至 0x1f）。

ddd 发送或接收字符字符，以八进制位指定。

\p 在处理前暂停 0.25 秒（仅用于发送）。

\d 处理前延时两秒（仅用于发送）。

\K 发送中断（零位的 0.25 秒）。

\M 禁用挂起（设置为 CLOCAL 或 LNOHANG）。

\m 启用挂起（不设置为 CLOCAL 或 LNOHANG）（缺省值）。

\c 请勿在发送前置字符串后添加回车字符（仅用于传送）。

\q 请勿在任何调试或登录输出中打印下列设置字符串（如口令）。后续 **\q** 序列开关“无提示”模式。

\A 将键入字符串分析为 IP 地址，用句点分隔为四个十进制数，用于本地终端的点对点连接（仅用于发送）。

举例

如下示例中，使用 Telebit PEP 调制解调器调用主机“everyone”，其 DTE 接口设置为 19200 bps。使用 V.32/V.42/V.42bis 调制解调器、具有运行 38400 DTE 能力的主机称为“nobody”，通过运行异步 PPP 的附加到 **/dev/ttya** 的直接电缆连接主机“someone”。通过附加到 SnapLink 端口 0 使用 T1 CSU/DSU 与“anyone”对话。通过 PPP 连接通道的 TCP 数据流将 realone.somewhere.com 的端口 77 连接到伪装主机。

如果没有成功连接到“someone”，两秒后重试。如果该尝试失败，下次尝试前等待 4 秒，然后为 8 秒、16 秒、32 秒，最后为 40 秒。持续每隔 40 秒尝试连接“someone”。“everyone”和“nobody”的重试时间间隔和最大回退缺省值为“10-3600”。

"" "" 表示法表明不期待任何内容，然后发送空白（其后为回车）。隐式回车通常在从远程系统引出响应时是有用的。

```
#
#       Systems - PPP systems file
#
```

everyone Any ACU 19200-PEP 5551212 in:--in: Pwe word: \qfoObar
nobody Any ACU 38400 5551213 in:--in: Pthey word: \qbaZz1ng
someone Any;2-40 cua 38400 0 in:--in: Pthem word: \qmeumBle
anyone Any rsd0a/0 1536000
pseudo-one Any;2-2 tcp/realone.somewhere.com/57

建议

缺省重试时间和回退（如 Any;10-3600）在使用拨号连接时正确，此时 PPP 连接必须在中断后尽快重新建立，但并不希望持续重新拨号到已经关闭的主机。如果两个系统间有专线，或者调用尝试不占用资源，使用更短的最大延时会更合适。

中等调用重试时间，如 60 秒，在使用拨号调制解调器建立双向连接的系统上运行正常，用于避免出现两端同时呼叫对方从而出现电话线路繁忙信号的僵局。由于源和应答调制解调器的不同行为，60 秒的时钟周期通常不同时开始，允许一端在没有干涉的情况下呼叫另一端。此外，可以在连接的两端设置不同的呼叫重试次数，有助于防止系统同步呼叫对方。

如果指定了主机名，请确认其地址在本地可用，即使在无连接时。如果必须启动连接以解析域名，请考虑在 **Filter** 和 **Systems** 两端使用主机 IP 地址替代（用句点分隔的十进制数）。

在有多个调制解调器或多种连接方法的系统间，可以安排自动失败恢复。如果两个系统通过专用线路连接（同步或异步），该条目首先应当为 **Systems** ，其后为另一条目，说明根据需要的拨号连接。有关详细信息，请参阅《 HP PPP User Guide》。

安全考虑

/etc/ppp/Systems 文件应该为模式 600。

作者

ppp.Systems 由 Progressive Systems 开发。

另请参阅

ppp.Auth(4)、 ppp.Services(4)、 ppp.Dialers(4)、 ppp.Filter(4)、 ppp.Keys(4)、 pppd(1)、 RFC 1548、 RFC 1332、 RFC 1144、 RFC 1055。

名称

privgrp - 特权值的格式

概要

#include <sys/privgrp.h>

说明

setprivgrp() 设置特权的掩码，而 *getprivgrp*(2) 返回一个结构数组，该数组给出在每一个组 ID 基础上的特权组赋值（请参阅 *getprivgrp*(2)）。 **setprivgrp()** 将一个内核功能与一个组 ID 相关联。这就允许将类似于超级用户的特权转分给一个特定组或一些特定组的成员。这些系统调用所需的常量或结构在 **<sys/privgrp.h>** 中定义。

各特权如下:

PRIV_RTPRIO	允许访问 **rtprio()** 系统调用（ *rtprio*(2)）。
PRIV_MLOCK	允许访问 **plock()** 系统调用（ *plock*(2)）。
PRIV_CHOWN	允许访问 **chown()** 系统调用（ *chown*(2)）。
PRIV_LOCKRDONLY	允许使用 **lockf()** 系统调用在为只读操作而打开的文件上设置锁（请参阅 *lockf*(2)）。
PRIV_SETRUGID	允许使用 **setuid()** 和 **setgid()** 系统调用分别更改一个进程的实际用户 ID 和实际组 ID（请参阅 *setuid*(2)）。
PRIV_MPCTL	允许使用 **mpctl()** 系统调用更改处理器绑定、位置域绑定或进程的启动策略（请参阅 *mpctl*(2)）。
PRIV_RTSCHED	允许访问 **sched_setparam()** 和 **sched_setscheduler()**，以设置 POSIX.4 实时优先级（请参阅 *rtsched*(2)）。
PRIV_SERIALIZE	允许使用 **serialize()** 强制目标进程与该系统调用所标记的其他进程按串行方式运行（请参阅 *serialize*(2)）。
PRIV_SPUCTL	允许在即时按需增容 (iCOD) 产品中进行专用的管理操作来停用或重新启用处理器。有关详细信息，请参阅该产品的文档资料。
PRIV_FSSTHREAD	允许在进程资源管理器 (PRM) 产品中进行专用的管理操作。有关详细信息，请参阅该产品的文档资料。
PRIV_PSET	允许更改系统 pset 配置（请参阅 *pset_create*(2)）。

特权在多字掩码中描述。每一个特权的 **#define** 值作为一个位索引进行解释（从 1 开始计数）。因此，通过将不同的位与掩码进行或运算，一个组 ID 可有与之关联的多个不同特权。

用指定的最大数量的具有专用特权的组配置系统。 **PRIV_MAXGRPS** 定义此最大数。在此最大数量的组中，有一个组是为全局特权而保留的（已授予给所有进程），而剩下的组可分配给实际的组 ID。

PRIV_MASKSIZ 定义多字掩码的大小，这些掩码在定义与组 ID 相关联的特权时使用。

从 **getprivgrp()** 系统调用中，以 **struct privgrp_map** 类型的结构数组形式向用户返回特权。该结构将多字掩码与一个组 ID 相关联。 privgrp_map 结构包含如下字段：

gid_t **priv_groupno**
uint32_t **priv_mask[PRIV_MASKSIZ]**

其中， *priv_groupno* 包含组 ID（请参阅 *setprivgrp*(2)），而 *priv_mask* 包含与 *priv_groupno* 相关联的特权掩码。

另请参阅

getprivgrp(1)、 setprivgrp(1M)、 chown(2)、 getprivgrp(2)、 lockf(2)、 plock(2)、 rtprio(2)、 rtsched(2)、 serialize(2)、 setgid(2)、 setuid(2)、 shmctl(2)、 mpctl(2)、 pset_create(2)。

名称

profile - 登录时设置用户环境

说明

如果文件 **/etc/profile** 存在，则每个登录的用户都会通过 Shell 执行此文件。文件 **/etc/profile** 应该进行设置来进行那些系统上的 *every* 用户希望的操作，或者设置合理的缺省值。如果用户登录（主）目录包含名为 **.profile** 的文件，则在该会话开始之前执行该文件（通过 Shell 的 **exec .profile** ）。 **.profile** 文件用于设置各种环境参数、设置终止模式或者覆盖执行 **/etc/profile** 的部分或全部结果。

举例

以下是典型的例子（除了注释）：

```
# 使一些环境变量成为全局变量
  export MAIL PATH TERM
# 设置文件创建掩码
  umask 22
# 告诉我新邮件何时进入
  MAIL=/var/mail/myname
# 将我的 /bin 目录添加到 Shell 搜索序列中
  PATH=$PATH:$HOME/bin
# 设置终端类型
  echo "terminal:\c"
  read TERM
  case $TERM in
    300)     stty cr2 nl0 tabs; tabs;;
    300s)    stty cr2 nl0 tabs; tabs;;
    450)     stty cr2 nl0 tabs; tabs;;
    hp)      stty cr0 nl0 tabs; tabs;;
    745|735) stty cr1 nl] -tabs; TERM=745;;
    43)      stty cr1 nl0 -tabs;;
    *)       echo "$TERM unknown";;
  esac
```

更完整的模型 **.profile** 可以在 **/etc/skel/.profile** 中找到。

文件

$HOME/.profile
/etc/profile

另请参阅

env(1)、 login(1)、 mail(1)、 sh(1)、 stty(1)、 su(1)、 environ(5)、 term(5)。

名称

proto: .proto - at 和 batch 的原型作业文件

概要

/var/adm/cron/.proto

/var/adm/cron/.proto. *queue*

说明

当一个作业提交给 **at** 或 **batch** 时，该作业构建为 POSIX shell 脚本（请参阅 *at*(1)）。作业文件在 **/var/spool/cron/atjobs** 中创建如下：

- **at** 创建一个将该作业作为 **at** 作业或 **batch** 作业描述的标题。 **at** 作业提交给除了队列 **a** 以外的所有队列，该队列作为 **batch** 作业列出。标题有：

  : at job　　对于 **at** 作业，或者

  : batch job　　对于 **batch** 作业。

- 添加 POSIX Shell 命令集来使 **at** 作业的环境（请参阅 *environ*(5)）与当前环境相同。
- 然后 **at** 将原型文件中的文本复制到作业文件，对于被其他文字替换的特殊 *variables* 除外：

  $d　　被当前工作目录替换。

  $l　　被当前文件大小限制（请参阅 *ulimit*(2)）替换。

  $m　　被当前 umask（请参阅 *umask*(2)）替换。

  $t　　被该作业应该运行的时间替换，表示为国际标准时间 1970 年 1 月 1 日 00:00 以后的秒数，之前有一个冒号。

  $<　　被 **at** 从标准输入读取的文本替换（即，提供给要在该作业中运行的 **at** 的命令）。

- 当作业提交给队列 *queue* 时， **at** 使用文件 **/var/adm/cron/.proto.***queue* 作为原型文件（如果存在的话）。否则它使用文件 **/var/adm/cron/.proto** 。

举例

下面的 **.proto** 文件创建命令来将当前目录、文件大小限制和该作业中的 umask 变成当 **at** 最初运行时相应的值。这些命令插入该作业的命令之前：

```
cd $d
ulimit $l
umask $m
$<
```

另请参阅

at(1)、 queuedefs(4)。

符合的标准

proto：SVID2、SVID3

名称

protocols - 协议名称数据库

说明

此文件将正式的协议名称和别名与协议编号关联。这允许用户通过一个符号名称而不是编号引用协议。对于每个协议，应当有一行带有如下信息：

<official protocol name> *<official protocol number>* *<aliases>*

这些映射在 RFC 1700 已分配的编号中定义。

别名是也可用于标识协议的其他名称。例如：

tcp 6 TCP

在此示例中，库调用 **getprotobyname()** 可以如下方式调用：

p = getprotobyname("TCP");

而不是

p = getprotobyname("tcp");

都生成相同的结果。

行不能以空格开头。项目通过任意数量的空白字符和（或）制表字符分隔。 **#** 字符指示注释的开始。搜索文件的例行程序不会解释从 **#** 开始直到行的结束的字符。

协议名可能包含除空格、换行或注释字符之外的任意可打印字符。结尾的空白字符或制表符允许在行尾。

举例

tcp 6 TCP # transmission control protocol
udp 17 UDP # user datagram protocol

作者

protocols 由加州大学伯克利分校开发。

文件

/etc/protocols

另请参阅

getprotoent(3N)。

名称

prpwd - 受保护的口令验证数据库文件用于信任系统

概要

/tcb/files/auth/*

说明

对验证配置文件的维护对应于系统中的每一个用户。用户配置文件保存在受保护口令数据库文件中，该数据库文件只有系统管理员可以访问。受保护口令数据库文件还包含了对应于用户帐户的加密口令。在信任系统中，口令对普通用户隐藏。

受保护口令数据库文件保留了对 **/etc/passwd** 和 **/etc/group** 文件的需要。为了使用系统，必须在 **/etc/passwd** 文件中进行定义用户。用户的受保护口令数据库文件包含用户名和用户 ID，用以提供与用户的 **/etc/passwd** 条目的关联关系。两者必须匹配，否则用户帐户将视作无效。

受保护口令数据库文件在 **/tcb/files/auth** 层次结构中进行维护。该目录包含了其他目录，每个目录以字符集中的单个字母命名。用户验证配置文件存储在基于用户帐户名首字母的目录中。这使得高效的搜索操作能够实现，用以快速定位到特定用户名的文件。例如，帐户 **root** 的验证配置文件存放在 **/tcb/files/auth/r** 目录中，可以通过打开 **/tcb/files/auth/r/root** 文件进行访问。

文件中定义的字段是用户特定值。这些值覆盖了系统缺省值。在使用系统缺省值之前，信任程序首先校验用户特定参数是否存在。

受保护的口令数据库文件包含取决于字段类型的关键字段标识符，以及字段值（某些字段类型不要求明确的值）。有关字段规范准确语法的描述，请参阅 *authcap*(4) 。字段规范与所有系统验证数据库相兼容。由受保护口令数据库文件支持的关键字字段标识符及其相关功能在以下描述中给出：

u_name 这是帐户的用户名，该帐户必须匹配文件名和 **/etc/passwd** 条目中相应的用户名。

u_id 这是帐户的用户 ID，必须与对应的 **/etc/passwd** 条目中的用户 ID 字段相匹配。

u_pwd 如果帐户有口令，则本字段包含了帐户的加密口令。

u_owner 本字段包含了帐户的所有者。

u_booauth 如果有这个字段并且包含一个大于 0 的值（通常为 1），同时在系统缺省文件中设置了启动验证标志，则该用户具有引导系统的权限。如果系统缺省文件中没有设置启动验证标志，则不使用该字段。

u_audid 本字段包含了用户审计 ID。

u_auditflag 本字段包含了用户审计标志。

u_minchg 本字段指定了更改口令的最小时间，以秒为单位。如果为非零值，自最后一次成功修改口令达到了指定的秒数后，口令才能更改，除非修改口令的用户被授权覆盖此限制。

u_maxlen 本字段指定了用户帐户口令的最大长度，应小于由 **<prot.h>** 中的常量 **AUTH_MAX_PASSWD_LENGTH** 定义的系统级最大值。

u_exp 本字段是一个 *time_t* 值，指定了帐户口令的到期时间。当口令到期，用户登录系统时，系统验证程序将要求更改口令。如果口令在更改之前过期，则帐户将被锁定。

u_life 本字段是一个 *time_t* 值，指定了口令的生命周期。如果到达生命周期，帐户将锁定，并且只能由授权的系统管理员解锁。

u_succhg 本字段是一个 *time_t* 值，用于指示最后一次成功修改口令的时间。本字段应当只通过能用于修改帐户口令的程序进行设置。

u_unsucchg 本字段是一个 *time_t* 值，用于指示最后一次口令修改失败的时间。本字段应当只通过能用于修改帐户口令的程序进行设置。

u_acct_expire 本字段是一个 *time_t* 值，指定了帐户的有效时间，以秒为单位。有效时间间隔过后，将不允许用户登录。本字段与 **u_life** 字段的区别在于， **u_life** 字段是根据上一次口令修改的时间来计时，而 **u_acct_expire** 字段不受口令更改的影响。

u_max_llogin 本字段值以秒为单位，指定相邻登录之间允许间隔的最大时间。如果上一次登录与当前时间之间的间隔时间超过该字段值，帐户将锁定，用户不能登录。

u_pw_expire_warning
本字段值以秒为单位，指定用户口令过期之前显示警告信息的时间。

u_pickpw 本字段值是一个标志，用于控制用户选择帐户口令的能力。它允许对帐户进行配置，使得用户不能选择口令，而只能使用系统为帐户生成的口令。

u_genpwd 本标志字段控制用户生成帐户口令的能力。系统具有生成口令的能力，口令可包含任意字母、字符或单词。

u_restrict 本标志字段控制是否对用户所选的任何口令，执行口令不重要性校验。执行不重要性校验，包括验证口令不代表登录名或组名，不代表回文（指顺读和倒读都一样的词语）或 *spell*(1) 程序识别的单词。有关口令不重要性校验的详细信息，请参阅 *acceptable_password*(3) 。

u_nullpw 本标志字段控制用户选择空的帐户口令的能力。

u_pwchanger 如果上一次帐户口令修改人非本帐户用户，则本字段记录了口令修改者的用户 ID。它用于警告登录用户，在用户不知情的情况下，帐户口令可能已被改变。

u_pw_admin_num
本字段包含了系统管理员重置帐户后，用户登录时必须提供的任意数字。登录成功后，本字段被清除。

u_genchars 本标志字段控制用户生成任意口令字符的能力。

u_genletters 本标志字段控制用户生成任意口令字母的能力。

u_tod 本字段包含了用于控制用户帐户何时能够登录的条目列表，条目指定格式均为一天中的时间，条目之间以逗号分隔。有关列表条目格式的详细信息，请参阅 *tod*(3) 。

u_suclog 本字段是一个 *time_t* 值，它包含了上一次成功登录到帐户的系统时间。

u_unsuclog 本字段是一个 *time_t* 值，它包含了上一次登录帐户失败的系统时间。

u_suctty 本字段是一个字符串，用于标识与上一次成功登录到帐户相关联的终端或远程主机的名称。

远程主机指定格式由表示主机 Internet 地址的 ASCII 码组成。使用 *gethostbyaddr*(3) ，将本字段转换为 Internet 地址和主机名。

u_numunsuclog 本字段包含了尝试登录到帐户失败的次数。当登录帐户成功时，本字段被重置。

u_unsuctty 本字段是一个字符串，用于标识与上一次试图登录帐户失败相关联的终端或远程主机的名称。

u_maxtries 本字段指定了在帐户被锁定前，尝试登录帐户连续失败的最大允许次数。

u_lock 本标志字段用于管理帐户锁定。用户不能登录到锁定的帐户。

注释

例行程序 *getprpwent*(3) 用于将受保护口令数据库文件解析为程序可用的数据结构。结构中的标志表明是否结构中的特殊字段以此定义。结构中也提供了系统缺省值。这些值来自于 **/tcb/files/auth/system/default** 字段，在没有用户指定值的情况下能被程序使用。

举例

以下是典型的受保护口令数据库文件的示例：

```
perry:u_name=perry:u_id#101:\
    :u_pwd=aZXtu1kmSpEzm:\
    :u_minchg#0:u_succhg#653793862:u_unsucchg#622581606:u_nullpw:\
    :u_suclog#671996425:u_suctty=tty1:\
    :u_unsuclog#660768767:u_unsuctty=tty1:\
    :u_maxtries#3:chkent:
```

这是关于 **perry** 的受保护口令数据库文件。 **perry** 的用户 ID 是 101。该值必须与 **/etc/passwd** 中的用户条目相匹配。帐户有一个口令，加密口令格式由 **u_pwd** 字段指定。

该数据库文件将口令更改的最小时间指定为 0，表明能在任何时候更改口令。此外，允许帐户有一个空口令 (**u_nullpw**) 。该帐户连续登录失败的最多次数为 3 次，表明 3 次登录失败后，帐号将锁定 (**u_maxtries**) 。其余字段提供了一些帐户信息，比如，最后一次更改口令成功和失败的时间、最后一次登录成功和失败的时间以及终端名称。

作者

prpwd 由 HP 开发。

另请参阅

login(1)、 users(1)、 acceptable_password(3)、 getprpwent(3)、 tod(3)、 authcap(4)、 default(4)。

名称

publickey - 公用密钥数据库

概要

etc/publickey

说明

/etc/publickey 是用于安全 RPC 的本地公用密钥数据库。可以选择 **/etc/publickey** 文件是否与其他公用密钥数据库一起使用，包括 NIS 公用密钥映射和 NIS+ 公用密钥映射。数据库中的每个条目由网络用户名（可能引用用户或主机名）组成，后接用户的公用密钥（十六进制的表示法）、一个冒号，然后是用户使用口令（也是十六进制的表示法）加密的私用密钥。

/etc/publickey 文件包含 **nobody** 的缺省条目。

作者

publickey 由 Sun Microsystems, Inc. 开发。

另请参阅

chkey(1)、 newkey(1M)、 getpublickey(3N)、 nsswitch.conf(4)。

名称

queuedefs - at、batch 和 crontab 的队列描述文件

概要

/var/adm/cron/queuedefs

说明

queuedefs 文件描述由 **cron** 管理的队列特性（请参阅 *cron*(1M) ）。此文件中的每个非注释行描述一个队列。行的格式如下:

*q***.**[*njob* **j**] [*nice* **n**] [*nwait* **w**]

此行中的字段是:

q 队列的名称， **a** 是由 **at** 启动的缺省作业队列（请参阅 *at*(1) ）， **b** 是由 **batch** 启动的作业队列（请参阅 *at*(1) ），而 **c** 是从一个 **crontab** 文件中（请参阅 *crontab*(1) ）运行的作业队列。在 **d** 到 **y** 范围内的队列名称指定用户定义的队列。

njob 在队列中可同时运行的最大作业数。尽管在此可指定任意数目，但在所有队列上可运行的作业总数限于 100 个。

nice 给队列上所有作业提供的 **nice** 值，这些作业都没有使用超级用户的用户 ID 运行（请参阅 *nice*(1) ）。缺省值为 2。

nwait 在重新调度一个被延迟的作业之前要等待的秒数，作业延迟的原因是多于 *njob* 个的作业正运行在作业队列中，或所有队列中有多于 100 个作业正在运行（请参阅上面的 *njob* ）。

举例

请考虑下述 **queuedefs** 文件:

a.4j1n
b.2j2n90w

文件解释如下:

a.4j1n 对于 **at** 作业的 **a** 队列（请参阅 *at*(1) ），可最多同时运行 4 个作业。这些作业将以 1 的 **nice** 值运行。

由于没有给定 *nwait* 值，如果一个作业因为正在运行的其他作业过多而无法运行，则 **cron** 将在再次尝试运行该作业之前等待 60 秒（请参阅 *cron*(1M) ）。

b.2j2n90w 对于 **batch** 作业的 **b** 队列（请参阅 *at*(1) ），可最多同时运行 2 个作业。那些作业将运行，且 **nice** 值为 2。如果一个作业因为正在运行的其他作业过多而无法运行，则 **cron** 将在再次尝试运行该作业之前等待 90 秒。

所有其他队列可最多同时运行 100 个作业。这些作业将运行，且 **nice** 值为 2。如果一个作业因为正在运行的其他作业过多而无法运行，则 **cron** 将在再次尝试运行该作业之前等待 60 秒。

另请参阅

at(1)、 nice(1)、 crontab(1)、 cron(1M)、 proto(4)。

符合的标准

queuedefs：SVID2、SVID3

名称

rc.config、rc.config.d - 包含系统配置信息的文件

概要

/etc/rc.config

/etc/rc.config.d/*

/etc/TIMEZONE

说明

在启动时使用的系统配置包含在目录 **/etc/rc.config.d** 中的文件中。文件 **/etc/rc.config** 执行 **/etc/rc.config.d** 和 **/etc/TIMEZONE** 中的所有文件，并将它们的内容导出到环境中。

/etc/rc.config

/etc/rc.config 文件是一个脚本，它执行所有 **/etc/rc.config.d/*** 脚本，而且还执行 **/etc/TIMEZONE** 。要读取配置定义，仅需要执行此文件。不管 **/sbin/rc** 什么时候运行，比如当运行 **init** 命令以在运行状态间转换时，它都会执行此文件。不管执行哪些启动脚本，位于 **/etc/rc.config.d** 中的每一个文件都会被执行。

/etc/rc.config.d

配置信息被构建为文件的一个目录，而不是一个包含相同信息的文件。这就允许开发人员在这里创建和管理他们自己的配置文件，而不会使共享所有权和访问通用文件复杂化。

/etc/rc.config.d/auditing

这是包含配置变量赋值的文件所在的位置。

必须编写由 POSIX Shell 而非 Bourne Shell、 **ksh** 或 **csh** 读取的配置脚本。在一些情况下，这些文件也必须由 **sd** 控制脚本或 *sam* 程序读取，并有可能由它们来修改。请参阅 *sd*(4) 和 *sam*(1M) 。由于这个原因，每一个变量定义必须出现在一个单独的行中，并使用语法：

variable=*value*

变量定义行中不能出现尾部注释。注释语句必须在单独的行中，且在第一列中使用 **#** 注释字符。此示例显示配置文件所需的语法：

```
# Cron configuration. See cron(1M)
# Cron configuration. See cron(1M)
#
# CRON: Set to 1 to start cron daemon
#
CRON=1
```

当描述变量配置的多个实例时，配置变量可声明为数组参数。例如，一个系统可能包含两个网络接口，每一个接口都有一个唯一的 IP 地址和子网掩码（请参阅 *ifconfig*(1M) ）。这样的声明示例如下：

```
NET_CARDS=2
IP_ADDRESS[1]=15.1.55.2
```

SUBNET_MASK[1]=255.255.248.0

IP_ADDRESS[2]=15.1.55.3
SUBNET_MASK[2]=255.255.248.0

注意，必须对执行的文件顺序没有要求。这就意味着配置文件禁止指向在其他配置文件中定义的变量，因为不能保证正被引用的变量当前已定义。在这些配置文件中，不能避免环境变量命名空间冲突。程序员必须注意避免这样的问题。

/etc/TIMEZONE

文件 **/etc/TIMEZONE** 包含 **TZ** 环境变量的定义。此文件是 POSIX 所必需的。在 **/etc/rc.config.d/*** 文件被执行的时候， **/sbin/rc** 执行此文件。

另请参阅

rc(1M)。

名称

rcsfile - RCS 文件格式

说明

RCS 文件是一个 ASCII 文件。其内容由下面的语法描述。该文本为任意格式，即，除非在字符串中，否则空格、制表符和换行符没有意义。字符串括在 @ 符号中。如果一个字符串包含 @ 符号，则该符号必须是双写的。

meta 语法使用下列约定：

|	（竖线） 分隔重叠。
{...}	（花括号） 括起可选短语。
{...}*	括起可能重复零次或多次的短语。
{...}+	括起必须至少出现一次并可能重复的短语。
<...>	括起非终端。

RCS 文件语法

标识符区分大小写。关键字只能小写。关键字集和标识符集可重叠。

```
<rcstext>   ::= <admin> {<delta>}* <desc> {<deltatext>}*

<admin>     ::= head    {<num>};
            access  {<id>}*;
            symbols {<id> : <num>}*;
            locks   {<id> : <num>}*; {strict ;}
            comment {<string>};

<delta>     ::= <num>
            date    <num>;
            author  <id>;
            state   {<id>};
            branches {<num>}*;
            next    {<num>};

<desc>      ::= desc    <string>

<deltatext> ::= <num>
            log     <string>
            text    <string>

<num>       ::= {<digit>{.}}+

<digit>     ::= 0 | 1 | ... | 9
```

<id> ::= <letter>{<idchar>}*

<letter> ::= A | B | ... | Z | a | b | ... | z

<idchar> ::= Any printing ASCII character except space,
tab, carriage return, newline, and <special>.

<special> ::= ; | : | , | @

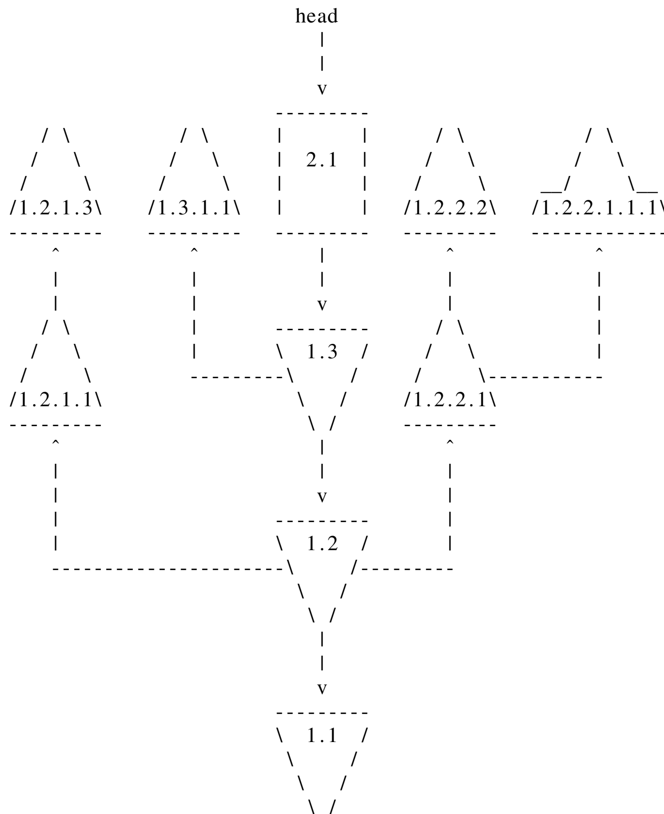
<string> ::= @{any ASCII character, with "@" doubled}*@

RCS 文件结构

<delta> 节点形成树状结构。其编号由一个单数字对组成的所有节点（例如 2.3，2.1，1.3 等）位于主干上，并通过 **next** 字段按数字的降序链接。 **<admin>** 节点中的 **head** 字段指向该序列的头节点（即，包含最高的数字对）。

其编号包含 2*n* 个字段 (*n*>=2) （例如，3.1.1.1，2.1.2.2 等）的 **<delta>** 节点链接如下。其前 (2*n*)-1 个数字段相同的所有节点，通过 **next** 字段按数字递增顺序链接。对于每个这样的序列，其数字与序列中 delta 版的前 2(*n*-1) 个数字段相同的 **<delta>** 节点称作分支点。节点的 **branches** 字段包含所有序列的第一个节点的数字列表，该列表就是这些序列的一个分支点。此列表按数字升序排列。

举例



警告

RCS 设计为仅与文本 (ASCII) 文件一起使用。 RCS 与非文本（二进制）文件一起使用将会导致数据损坏。

作者

rcsfile 由西拉斐特 Purdue 大学的 Walter F. Tichy 于 47907 开发。修订编号：3.0. 发行时间：83/05/11. 版权所有 1982 Walter F. Tichy。

另请参阅

ci(1)、 co(1)、 ident(1)、 rcs(1)、 rcsdiff(1)、 rcsmerge(1)、 rlog(1)、 rcsintro(5)。

名称

resolver: resolv.conf - 解析程序配置文件

概要

/etc/resolv.conf

说明

解析程序是 C 库中的例行程序集，（请参阅 *resolver*(3N)），提供对 Internet 域名系统的访问。解析程序的配置文件包含第一次被进程调用时由解析例行程序读取的信息。文件设计为用户可读，包含关键字和值的列表，提供各种类型的解析程序信息。

如果查询的唯一名称服务器在本地计算机上，则此文件并不总是必需的。如果设置为完全限定域名，域名可由主机名确定（请参阅 *hostname*(1)）。

可以识别的配置选项包括：

nameserver 解析程序查询的名称服务器 Internet (IP) 地址，以点分法标识。最多可列出 **MAXNS** 个名称服务器（当前为 3），每个服务器有一个关键字。如果有多个服务器，解析程序库以列出的顺序对其进行查询。如果没有 **nameserver** 条目存在，缺省使用本地计算机上的名称服务器。（使用的算法为：尝试某个名称服务器，如果超时，尝试下一个，持续直至尝试了全部名称服务器，然后重新尝试全部名称服务器，直至达到最大重试次数）。

domain 本地域名。大多数域内的名称查询可以使用相对于本地域的短名称。如果没有 **domain** 条目存在，域从 **gethostname()** 返回的本地主机名称中确定（请参阅 *gethostname*(2)）；域部分解释为第一个点 (**.**) 之后的所有内容。最后，如果主机名不包含域部分，则假定为根域。

retrans 重新传输超时。在 **res_init()** 调用过程中解释（请参阅 *resolver*(3N)）。比通过 **set_resfeild()** （请参阅 *resolver*(3N)）API 设置优先级高，比通过 **RES_RETRANS** （请参阅 *resolver*(3N)）环境变量设置优先级低。为 **retrans** 指定了无效值时，在系统日志中标记信息。缺省值为 5000 毫秒。

要使用 **retrans** 选项，在 **/etc/resolv.conf** 中添加下列名称值对：

retrans *value-in-milliseconds*

例如：要将重新传输值设置为 6000，请使用：

retrans 6000

retry 重试次数。在 **res_init()** 调用过程中解释（请参阅 *resolver*(3N)）。比通过 **set_resfeild()** （请参阅 *resolver*(3N)）API 设置优先级高，比通过 **RES_RETRY** （请参阅 *resolver*(3N)）环境变量设置优先级低。为 **retry** 指定了无效值的时候，在 **syslog** 中标记信息。缺省值为 4。

要使用 **retry** 选项，在 **/etc/resolv.conf** 中添加下列名称值对：

retry *number-of-retries*

例如：要将重试次数设置为 6，请使用：

retry 6

search 搜索列表查找主机名。如果没有使用搜索选项，搜索列表将仅包括本地域名。搜索列表可以通过列出所需域名搜索路径进行更改，该路径后跟随 **search** 关键字，使用空格或制表符与名称分隔。大部分解析程序将尝试依次使用搜索路径的每个组成部分进行查询，直至找到匹配项。请注意列出的域不在本地时，搜索过程可能很慢，并会生成大量的网络流量，如果对于一个域没有服务器可用，则查询超时。

当前搜索列表被限制为六个域，最大 256 个字符。

为了保证在不同文件中短文件名正常工作，搜索列表中的第一个域必须是本地域（例如 **.rhosts** 和 **inetd.sec** ）。

sortlist 使得 *gethostbyname*(3N) 返回的地址按地址列表中指定的网络编号排序。在接收本地名称服务器查询到的多个地址时，此选项允许为 **gethostbyname()** 指定首选的子网和网络。语法为

sortlist *addresslist*

IP 地址网络掩码对的排序列表。网络掩码为可选项，缺省值为网络的网络掩码。使用斜线分隔 IP 地址和可选网络掩码对。最多可以指定 10 对。使用空格分隔这些对。

下列排序列表指令在 128.32.42 子网上对地址排序。

sortlist 128.32.42.0/255.255.255.0

斜线后的参数是所述子网的子网掩码。要首选整个网络，用户可以省略斜线和子网掩码：

sortlist 128.32.0.0

gethostbyname() 将任何地址按照它们所匹配的参数在排序列表出现的顺序排列，并将不匹配的地址追加到末尾。

options 选项允许修改某些内部解析程序的变量。语法为

options *option* ...

当前选项支持如下：

ndots:*n* 设置点数的阈值，在启动绝对查询之前，此阈值必须出现在赋给 **res_query** （请参阅 *resolver*(3N) ）的名称中。 *n* 的缺省值是“1”，表明如果在名称中出现点，则此名称在追加任何搜索列表元素之前首先作为绝对名称尝试。

domain 和 **search** 选项是互相排斥的。如果出现了多个关键字实例，最后一个实例覆盖其他实例。

通过将环境变量 **LOCALDOMAIN** 设置为空格分隔的搜索域列表，系统 **resolv.conf** 文件的 **search** 关键字可以在每个进程的基础上被覆盖。如上文 **options** 所述，通过将环境变量 **RES_OPTIONS** 设置为空格分隔的解析程序选项列表，系统 **resolv.conf** 文件的 **options** 关键字可以针对各进程进行修改。

关键字和值必须出现在单独一行中，关键字（例如 **nameserver** ）必须在行的开始。关键字后面的值以空白字符分隔。

请注意解析例行程序 **res_init()** 在读取文件时默默忽略错误（请参阅 *resolver*(3N) ）。

举例

典型的 **resolv.conf** 文件类似于下面列举内容:

domain div.inc.com
nameserver 15.19.8.119
nameserver 15.19.8.197

警告

为了减少连接到非目的地的情况，管理员应当小心选择放入 **resolv.conf** 文件搜索列表中的域。 HP 建议将搜索列表中的可用域限制为用户信任组织所管理的域。有关安全完成搜索列表的详细信息，请参阅位于 **/usr/share/doc** 的 *RFC 1535* 。

作者

resolver 由加州大学伯克利分校开发。

文件

/etc/resolv.conf 解析程序配置文件。

另请参阅

named(1M)、 resolver(3N)、 gethostent(3N)、 hostname(5)、

RFC 1535。

名称

rmtab - 本地文件系统挂接统计信息

说明

文件 **/etc/rmtab** 包含从此计算机挂接远程文件系统的所有客户端记录。一旦远程 **mount** 完成，一个条目会在服务于该文件系统的计算机的 **rmtab** 文件中生成。 **umount** 删除远程挂接的文件系统的条目。 **umount -a** 广播至所有服务器，这些服务器应当从由广播消息的发送者创建的 **rmtab** 中删除所有条目。该表为一系列下列形式的行：

hostname:*directory*

该表仅保留故障之间的信息，而且仅在启动时通过 **mountd** 读取（请参阅 *mountd*(1M)）。 **mountd** 保留核心表以处理命令的请求，比如 **showmount** 和 **shutdown** （请参阅 *showmount*(1M) 和 *shutdown*(1M)）。

警告

尽管 **rmtab** 表接近正确，但是不完全准确。

作者

rmtab 由 Sun Microsystems, Inc. 开发。

文件

/etc/rmtab

另请参阅

mount(1M)、 mountd(1M)、 showmount(1M)、 shutdown(1M)。

名称

rndc.conf - rndc 配置文件

说明

用于控制名称服务器的 BIND 实用程序 **rndc** 有它自身的配置文件 **/etc/rndc.conf** 。这个文件在结构与句法上类似于命名的配置文件，即 **named.conf** 。语句括在括号中并以半角冒号终止。语句中的子句也用半角冒号终止。一般支持的注释格式为：

> **C style: /* */**
> **C++ style: // to end of line**
> **Unix style: # to end of line**

rndc.conf 文件的语法比 **named.conf** 更简单。此文件包括三条语句：

> **options statement**
> **server statement**
> **key statement**

options 语句包含两个子句： **default-server** 和 **default-key** 。

当执行 **rndc** 时如果服务器没有在命令行中指定，则 **default-server** 子句用于指定 **rndc** 运行的缺省服务器。**default-server** 关键字后面是名称服务器的名称或地址。

当执行命令行时，如果没用 **-y** 选项指定密钥，则 **default-key** 子句用于指定用来验证服务器的命令和响应的缺省密钥。 **default-key** 关键字后面是通过 **key** 语句标识的密钥名称。

server 语句以一个标识字符串、主机名或名称服务器的地址开头。此语句有一个单个子句，即 **key** 。密钥名称必须与在 **key** 语句中指定的名称匹配。

key 语句以一个标识字符串（即密钥的名称）开头。这个语句包含两个子句： **algorithm** 和 **secret** 。

algorithm 子句可标识 **rndc** 使用的加密算法。目前仅支持 HMAC-MD5。

secret 子句包含用于验证的随机密钥。 base-64 编码使用的算法在 **algorithm** 子句中指定。 base-64 字符串括在双引号中。

BIND 9 程序 **dnssec-keygen** 可用于生成 **secret** 子句的 base-64 字符串。

举例

如果主机和密钥的名称匹配于一个关键字，它们必须用双引号括起来，比如有一个“key”名称的密钥。

```
options {
     default-server  localhost;
     default-key     samplekey;
    };

server localhost {
```

```
        key samplekey;
       };

key samplekey {
        algorithm hmac-md5;
secret "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
        };
```

在上面的例子中， **rndc** 将缺省使用本地主机 (127.0.0.1) 的服务器和名称为 *samplekey* 的密钥。本地主机服务器的命令将用 *samplekey* 密钥。 **key** 语句指明 *samplekey* 使用 HMAC-MD5 算法并且它的 *secret* 子句包含括在双引号中的 HMAC-MD5 密钥的 base-64 编码。

可以用 **dnssec-keygen** 生成密钥，如下所示：

$ dnssec-keygen -a hmac-md5 -b 128 -n user rndc

base-64 字符串将出现在两个文件中， *Krndc.+157.+{random}.key* 和 *Krndc.+157.+{random}.private* 。提取置于 **rndc.conf** 和 **named.conf key** 语句中的密钥后，可以删除 **.key** 和 **.private** 文件。

名称服务器配置

使用 **named.conf** 中的 **controls** 语句，名称服务器必须配置为接收 **rndc** 连接并标识指定在 **rndc.conf** 文件中的密钥。

限定

目前没有方法指定 **rndc** 必须运行的端口。

作者

rndc.conf 由 ISC (Internet Software Consortium) 开发。

另请参阅

dnssec-keygen(1)、 rndc(1)、 named(1M)。

名称

rpc - rpc 程序编号数据库

概要

/etc/rpc

说明

文件 **/etc/rpc** 包含用户可读的名称，这些名称可用于代替 RPC 程序编号。每行包含如下信息：

- RPC 程序的服务器名称
- RPC 程序编号
- 别名

项目通过任意数量的空白字符和（或）制表字符分隔。文件中任何位置的 **#** 指明直到行结尾都为注释。

举例

下面是一个 **/etc/rpc** 文件的示例：

```
#
# rpc 12.0 89/09/25
#
rstatd          100001    rstat rup perfmeter
rusersd         100002    rusers
nfs             100003    nfsprog
ypserv          100004    ypprog
mountd          100005    mount showmount
ypbind          100007
walld           100008    rwall shutdown
yppasswdd       100009    yppasswd
etherstatd      100010    etherstat
rquotad         100011    rquotaprog quota rquota
sprayd          100012    spray
selection_svc   100015    selnsvc
dbsessionmgr    100016    unify netdbms dbms
rexd            100017    rex remote_exec
office_auto     100018    alice
```

作者

rpc 由 Sun Microsystems, Inc. 开发。

文件

/etc/rpc

另请参阅

getrpcent(3C)。

名称

rtradvd.conf - 路由器广播守护进程配置文件

说明

此文件描述了 *rtradvd*(1M) 在为某个特定接口创建 IPv6 路由器广播数据包时所使用的信息，有关创建数据包的信息如 RFC 2461（“Neighbor Discovery for IP Version 6”）所述。 **rtradvd** 会在初始化时或者接收到一个 **SIGHUP** 信号时读取此文件。

rtradvd 配置包含两个通用语句（ defaults 、 **interface** ）和一个子语句 (**prefixinfo**) ，子语句仅由 **interface** 语句使用。所有的语句和子语句都必需以分号结束。语句由空白字符分隔的关键字组成，空白字符可以是任意空格和制表符的组合。

用户可以为每个要启动路由器广播的接口创建一个条目。任何没有在单个 **interface** 或者 **prefixinfo** 条目中指定的配置信息都继承自 **defaults** 语句。如果没有 **defaults** 语句或者特定的关键字，选项的值会跟据 RFC-2461 中定义的缺省值进行设置。

defaults 语句可以用来为 **interface** 和 **prefixinfo** 关键字定义全局值。 **defaults** 语句定义为：

```
defaults {
  # list of interface specific keyword-value pairs;
  # list of prefixinfo specific keyword-value pairs;
};
```

interface 语句定义为：

```
interface interface-name {
  # list of interface specific keyword-value pairs;
  prefixinfo prefix/prefixlen {
    # list of prefixinfo specific keyword-value pairs;
  };
};
```

prefixinfo 指定了在该接口上发布公告的“前缀”和“前缀长度”。

interface 和 **prefixinfo** 特定关键字（如果指定的话）比定义在 **defaults** 语句中的关键字具有更高的优先级。

注释：注释起始于井号字符 *(#)* ，并在此行末尾结束。

接口特定选项

下面基本的关键字和值定义在 RFC 2461 中：

AdvSendAdvertisement on|off

启用 (**on**) 或者禁用 (**off**) 发送周期性的路由器公告或者响应路由器请求。

缺省：禁用

MaxRtrAdvInterval *integer*

指定了从此接口上主动发送的组播路由器广播报文之间的最大时间间隔，以秒为单位。有效的值在 4 和 1800 秒之间，包括该值。当使用移动 IPv6 扩展时，所允许的最小有效值降到 0.07。

缺省： 600

MinRtrAdvInterval *integer*

指定了从此接口上主动发送的组播路由器广播报文之间的最小时间间隔，以秒为单位。有效值在 3 秒和 .75 * **MaxRtrAdvInterval** 之间，包括该值。缺省值为 200 秒。当使用移动 IPv6 扩展时，所允许的最小有效值降至 0.03。

缺省： 0.33 * **MaxRtrAdvInterval**

AdvManagedFlag on|off

启用 (**on**) 或者禁用 (**off**) 路由器广播中“受管理的地址配置”标志的设置。

缺省：禁用

AdvOtherConfigFlag on|off

启用 (**on**) 或者禁用 (**off**) 路由器广播中“其他状态配置”标志的设置。

缺省：禁用

AdvLinkMTU *integer*

指定要放置在路由器发送的 MTU 选项中的值。取值为零表示未指定。当设置为 0， **rtradvd** 不包含路由器广播中的 MTU 选项。它不能大于一个特定接口最大允许的 MTU。

缺省： 0

AdvReachableTime *integer*

指定一个以毫秒为单位的时间，并放在路由器广播消息中的可达时间域中。有效值在 0 和 3,600,000（一小时）之间，包括此值。取值为零表示路由器未指定。

缺省： 0

AdvRetransTimer *integer*

指定一个以毫秒为单位的时间，并放置在路由器广播消息的重传计时器字段中。取值为零表示路由器未指定。

缺省： 0

AdvCurHopLimit *integer*

指定一个值，它将放置在路由器发送的路由器广播中的 Cur Hop Limit 字段中。 取值为零表示路由器未指定此值。

缺省： 64

AdvRouterLifetime *integer*

指定一个以秒为单位的时间，它会放置在路由器广播的路由器生命期字段中。有效值为 0 或者在 **MaxRtrAdvInterval** 和 9000 之间，包括此值。取值为零意味着路由器不会用做缺省路由器，并且不应该出现在接收路由器广播消息的主机的缺省路由器列表中。

缺省： 3 * **MaxRtrAdvInterval**

AdvSourceLLAddress on|off

在外发的路由器广播消息中启用 (**on**) 或者禁用 (**off**) 发送接口链路层地址选项。

缺省：启用

AdvHomeAgentFlag on|off

移动 IPv6 选项。启用 (**on**) 或者禁用 (**off**) 路由器广播消息中“本地代理”标志字段的设置。当启用时，表示发送此路由器广播消息的路由器同时作为此链路上的移动 IPv6 本地代理。当启用时，移动 IPv6 扩展所指定的最小值可用于 **MinRtrAdvInterval** 和 **MaxRtrAdvInterval** 。

缺省：禁用

AdvHomeAgentLifetime *integer*

移动 IPv6 选项。指定路由器提供移动 IPv6 本地代理服务的时间长度，以秒为单位。最大值为 65520 秒（18.2 小时）。不允许取值为 0。

缺省： **AdvRouterLifetime**

AdvHomeAgentPreference *integer*

移动 IPv6 选项。为发送此路由器广播的本地代理指定优先级。取值越高表示一个本地代理越优先使用。如果 **AdvHomeAgentPreference** 和 **AdvHomeAgentLifetime** 被设置为它们的缺省值，则本地代理信息选项不会包含在此本地代理路由器所发送的路由器广播消息中。

缺省： 0

AdvIntervalOpt on|off

移动 IPv6 选项。在路由器广播消息中启用 (**on**) 或者禁用 (**off**) 发送公告间隔选项。 如果此选项包含在消息中，可以使用移动 IPv6 扩展为 **MinRtrAdvInterval** 和 **MaxRtrAdvInterval** 指定的最小限制。

缺省：禁用

前缀特定选项

下列地址前缀关键字和值定义在 RFC 2461 中：

AdvValidLifetime seconds|forever

指定地址前缀的有效生存期，以秒为单位，它将放在外发路由器广播消息中。符号名称 **forever** 代表无限。

缺省：2592000 秒（30 天）

AdvPreferredLifetime seconds|forever

指定地址前缀的首选周期，以秒为单位，被放置于外发路由器广播消息中。符号名称 **forever** 代表无限。

缺省：604800 秒（7 天）

AdvOnlinkFlag on|off

在前缀信息选项中设置 (**on**) 或者清除 (**off**) On-link 标志字段。

缺省：设置

AdvAutonomousFlag on|off

在前缀信息选项中设置 (**on**) 或者清除 (**off**) Autonomous Flag 字段。

缺省：设置

AdvRouterAddress on|off

移动 IPv6 前缀选项。在前缀信息选项中设置 (**on**) 或清除 (**off**) 路由器标志字段。这表示前缀字段除了广播指定的前缀外，还包括一个指定给发送路由器完整的 IP 地址。

缺省：设置

举例

三个示例 **rtradvd** 配置文件如下所示，它们适用于有两个接口的系统。在这些例子中，如果配置文件中未指定那些关键字，守护程序会使用 RFC-2461 中所定义的缺省值。第二个例子使用 **defaults** 语句为接口和前缀定义了一些全局的关键字。最后，第三个例子显示了一个移动 IPv6 的例子。

示例 **1**

```
interface lan1
{
    AdvCurHopLimit              32;
    AdvSendAdvertisement    on;
    prefixinfo 2008:0:0:4::/64
    {
        AdvValidLifetime 70100;
        AdvPreferredLifetime 50200;
    };
    prefixinfo 2008:0:0:5::/64
    {
        AdvValidLifetime 70100;
        AdvPreferredLifetime 50200;
    };
};

interface lan2
```

```
{
    AdvCurHopLimit        32;
    AdvRouterLifetime     0;
    AdvSendAdvertisement  on;
    prefixinfo 2008:0:0:6::/64
    {
        AdvValidLifetime     70100;
        AdvPreferredLifetime 50200;

    };
};
```

示例 1 指定了守护进程应当在 **lan1** 和 **lan2** 上发送路由器广播消息。

AdvSendAdvertisement 必须是 **on** （缺省为 **off** ）。两个接口的 **AdvCurHopLimit** 都设置为 32（缺省值为 64）。第二个接口的 **AdvRouterLifetime** 设置为 0。这表示该接收主机不应当将此路由器添加到它们的“IPv6 缺省路由器”的列表中。

示例 **2**

```
defaults {
    AdvCurHopLimit        32;
    AdvSendAdvertisement  on;
    AdvValidLifetime      70100;
    AdvPreferredLifetime  50200;
};

interface lan1
{
    prefixinfo 2008:0:0:4::/64
    {

    };
    prefixinfo 2008:0:0:5::/64
    {
    };
};

interface lan2
{
    AdvRouterLifetime 0;
    prefixinfo 2008:0:0:6::/64
```

```
        {
        };
};
```

和示例 1 相同但使用了 **defaults** 语句。

示例 **3**

```
defaults {
        AdvSendAdvertisement    on;
        MinRtrAdvInterval       2;
        MaxRtrAdvInterval       5;
        AdvHomeAgentFlag        on;
        AdvHomeAgentLifetime    1200;
        AdvHomeAgentPreference  3;
        AdvIntervalOpt          on;
};

interface lan1
{

        prefixinfo 2008:0:0:4::/64
        {

        };
        prefixinfo 2008::5:210:83ff:fef7:7a9d/64
        {
                AdvRouterAddress on;
        };
};

interface lan2
{

        prefixinfo 2008:0:0:6::9/64
        {
                AdvRouterAddress     on;
                AdvPreferredLifetime 18000;
        };
};
```

在此例中，移动 IPv6 选项（ AdvHomeAgentFlag 、 **AdvHomeAgentLifetime** 、 **AdvHomeAgentPreference** ）会

包含在路由器广播消息中。 **AdvRouterAddress** 选项会设置为 **on** 将路由器的全局地址也在 **lan1** 和 **lan2** 上广播。如果检测到移动 IPv6 产品的内核部分（请参阅 *mip6mod(7)* ），这个节点也要将在 **lan1** 和 **lan2** 上配置为 **AdvRouterAddress** 的两个前缀作为“本地代理”。在这种情况下，守护进程能够处理四个新的由移动 IPv6 定义的 ICMPv6 消息（请参阅 *rtradvd(1M)*）。

另请参阅

rtradvd(1M)、ndp(1M)、mip6mod(7)、ndp(7P)

1. T. Narten， E. Nordmark， W. Simpson， 《Neighbor Discovery for IP Version 6 (IPv6)》， RFC2461， 1998 年 9 月。
2. S. Thompson， T. Narten， 《IPv6 Stateless Address Autoconfiguration》， RFC2462， 1998 年 9 月。
3. D. Johnson， C. Perkins， 《Mobility Support in IPv6》， IETF 文档。

名称

sccsfile - SCCS 文件的格式

说明

一个 SCCS 文件是一个包括六个逻辑部分的 ASCII 文件：

checksum	文件中除第一行外所有字符的总和。
delta table	包含每个 delta 版的信息。
user names	允许添加 delta 版的用户的登录名和（或）数字组 ID。
flags	内部关键字的定义。文件的 *comments* 任意描述信息。
body	混合着控制行的实际文本行。

贯穿 SCCS 文件并有以 ASCII SOH （标题开始）字符（八进制 001）开头的行。这个字符今后表示为 “控制字符”并用 @ 图形表示。下面介绍的不是描述为控制字符开头的任意行，将禁止以控制字符开头。在 SCCS 文件中的所有行的长度将限制为 **BUFSIZ** （定义为 **<stdio.h>** ） 个字符。

格式 *DDDDD* 的条目代表一个五位数（一个介于 00000 到 99999 之间的数字）。

下面详细地描述一个 SCCS 文件的每个逻辑部分：

Checksum 校验是 SCCS 文件的第一行。此行的形式如下：

@h*DDDDD*

校验的值是除了第一行外所有字符的总和。 **@h** 序列提供一个 “幻数”，由介于 0x01 到 0x68 的两个字节组成。（类似于 UNIX 操作系统的其他版本通常用这个相同的值，但它可能显示或记录为带有不同字节顺序的单个数字）。

Delta table delta 版表由可变数量的格式条目组成。

@s *DDDDD/DDDDD/DDDDD*
@d <*type*> <*SID*> *yr/mo/da hr:mi:se* <*pgmr*> *DDDDD DDDDD*
@i *DDDDD* . . .
@x *DDDDD* . . .
@g *DDDDD* . . .
@m <*MR number*>
.
.
.
@c <*comments*> . . .
.
.
.
@e

第一行 (**@s**) 分别包含插入/删除/未改变行的数字。第二行 (**@d**) 分别包含 delta 版的类型（当前为正常： **D** 和删除的： **R** ）、delta 版的 *SID* (SCCS ID)、 delta 版创建的日期和时间、 delta 版创建时对应的实际用户 ID 以及 delta 版与先前版的序列号。

@i 、 **@x** 和 **@g** 行分别包含、排除及忽略包括 delta 版在内的序列号。这些行是可选的。

每个 **@m** 行（可选的）包含一个与 delta 版有关的 MR （修改请求）编号； **@c** 包含与 delta 版有关的注释。

@e 行结束 delta 版表条目。

User names 允许将 delta 版添加到文件中的用户的登录名和（或）数字组 ID 的列表，以换行符分隔。包含这些登录名和（或）数字组 ID 的那些行要用括号行 **@u** 和 **@U** 括起来。一个空列表允许任何人创建 delta 版。以 **!** 开头的任何行禁止指定的组或用户创建 delta 版。

Flags 内部使用的关键字（有关它们使用的详细信息，请参阅 *admin*(1) ）。每个标志行采用的形式如下:

@f <*flag*> <*optional text*>

定义以下标志:

@f t <*type of program*>
@f v <*program name*>
@f i <*keyword string*>
@f b
@f m <*module name*>
@f f <*floor*>
@f c <*ceiling*>
@f d <*default-sid*>
@f n
@f j
@f l <*lock-releases*>
@f q <*user defined*>
@f z <*reserved for use in interfaces*>

上面标志的功能如下:

t 定义了 **%Y%** 标识关键字的替代。

v 控制除注释外的 MR 编号提示。如果提供可选文本，则它定义了一个 MR 编号-有效性检查程序。

i 控制“No id keywords”消息的警告/错误方面。没有 **i** 标志时，消息仅仅为警告；有 **i** 标志时，此消息导致一个致命错误（文件中的 *get* 失败,或者没有

创建 delta 版）。

b 有 **b** 标志时， **-b** 键字母可以使用 *get* 命令生成 delta 版的树的分支。

m 定义 **%M%** 标识关键字替换文本的第一选择。

f 定义“下限”版本；更低版本不能添加 delta 版。

c 定义“上限”版本；更高版本不能添加 delta 版。

d 当 *get* 命令中未指定任何内容时，定义要使用的缺省 SID 。

n 使得 *delta* 将“空”delta 版（一个应用 *no* 改变的 delta 版）插入到在一个 *new* 版本中创建 delta 版时跳过的那些版本中（例如，在 delta 2.7 后创建 delta 5.1 时，将跳过版本 3 和 4）。缺少 **n** 标志会导致跳过的版本全空。

j 使得 *get* 允许同一个基础 SID 的并发编辑。关于限制大信息，请参阅 *admin*(1) 。

l 定义一个版本的 *list* 是 *locked* 以防止编辑（*get*(1) 带有 **-e** 键字母）。

q 定义了 **%Q%** 标识关键字的替代。

z 用于某个特定的专用界面程序。

Comments 任意文本将括在括号行 **@t** 和 **@T** 中。典型注释段包含文件目的描述。

Body 由文本行和控制行组成。文本行并非以控制字符开头；控制行以控制字符开头。有三种控制行：

键入	表示为：
insert	**@I** *DDDDD*
delete	**@D** *DDDDD*
end	**@E** *DDDDD*

数字字符串是与控制行 delta 版有关的序列号。

警告

SCCS 文件可以是任意长度的，但是文本文件自身中的行数不能超过 99 999 行。

另请参阅

admin(1)、 delta(1)、 get(1)、 prs(1)。

名称

securenets - NIS 映射安全文件

说明

/etc/securenets 文件定义可能访问服务器上 NIS 映射的网络和主机。文件中的每行都给出一个网络掩码和一个网络地址，且它们都是点分格式。例如：

255.255.255.255 133.33.33.33

文件可能具有任意数量的网络掩码/网络对。

当 **ypserv** 在服务器上启动时，它检查 **/etc/securenets** 是否存在，若存在，则将其内容读入内存。 **ypserv** 必须停止并重新启动以使 **/etc/securenets** 内的任意变动生效。

启动时，网络掩码和网络地址被转换成二进制格式，并进行逻辑与处理。其结果必须等于网络地址（第二个地址）才可合法。

如果网络掩码是 255.255.255.255 （二进制表示时所有位为 1），则网络地址参数中的任意一个地址都与之相匹配。如果网络掩码中的任意一个字段为 0，则网络地址中的相应字段必须是 0。当以这种方式使用时，地址中指定为 0 的部分将用作一个通配符。

当一个客户端尝试绑定到服务器时， **ypbind** 对照在 **/etc/securenets** 文件中给定的 IP 地址检查客户端的 IP。同样，该地址被转换为二进制形式，并与网络掩码进行逻辑与处理。其结果必须等于在文件中指定的网络地址。如果客户端地址不匹配文件中的任意一个地址对，则绑定被拒绝，并产生消息“服务器的 NIS 域中无此映射”。

securenets 文件可用于限制对特定主机的访问，或使用通配符功能限制对子网的访问。

如果在 **/etc/securenets** 文件中存在语法错误，则消息会记录到 **ypserv** 日志文件中（缺省为 **/usr/adm/syslog** ），且 **ypserv** 不启动。

如果一个主机有多个接口，则每个接口地址必须在该主机的安全网络文件中具有可靠的 NIS 访问。

举例

/etc/securenets 中的这一行仅提供对地址为 192.33.33.33 的主机的访问：

255.255.255.255 192.33.33.33

此条目允许由 192.33.33 子网上的任意一个主机访问：

255.255.255.0 192.33.33.0

对于更大范围的访问，例如对于整个企业，此条目允许使用地址以“15”开头的任意主机：

255.0.0.0 15.0.0.0

另请参阅

ypserv(1M)

名称

security - 安全缺省配置文件

说明

许多系统命令和功能，根据 **/etc/default/security** 配置文件中的相关参数进行配置。此文件必须允许任何人读取和写入根。

文件中的每一行作为系统给定的命令或功能的注释或配置信息处理。如果一行的开头为 **#** ，则此行为注释行。无注释的行格式为 **parameter=value** 。

如果任何参数没有定义或注释出了该行，下面详细列举要应用的缺省行为。

参数定义、有效值和缺省值定义如下：

ABORT_LOGIN_ON_MISSING_HOMEDIR

此参数在用户主目录不存在的情况下控制登录行为。请注意此项仅用于非根用户，并仅应用于 *login*(1) 命令或不直接调用 *login*(1) 的服务，如 *telnetd*(1M) 和 *rlogind*(1M) 命令。

ABORT_LOGIN_ON_MISSING_HOMEDIR=0

如果用户主目录不存在，使用“/”作为主目录登录。

ABORT_LOGIN_ON_MISSING_HOMEDIR=1

如果用户主目录不存在，退出登录会话。

缺省值： **ABORT_LOGIN_ON_MISSING_HOMEDIR=0**

BOOT_AUTH

此参数控制将系统启动为单用户模式时是否需要验证。如果启用，只有在验证用户提供的口令后系统方可启动进入单用户模式。此参数不用于信任系统。但是，如果在标准系统上启用了引导验证，则当系统转换为信任系统时，引导验证同样也作为信任系统的缺省值启用。

BOOT_AUTH=0

关闭引导验证。

BOOT_AUTH=1

启用引导验证。

缺省值： **BOOT_AUTH=0**

BOOT_USERS

此参数定义从控制台以单用户模式引导系统的验证用户名。名称用逗号分隔 **(,)** 。仅在启用引导验证时有效。请参阅 **BOOT_AUTH** 参数说明。 **BOOT_USERS** 参数不应用于信任系统。但是，当标准系统转换成为信任系统时，此信息被转换。

BOOT_USERS=*mary,jack*

除根用户之外，用户 *mary* 或 *jack* 均可以将从控制台引导系统至单用户模式。

缺省值： **BOOT_USERS=root**

MIN_PASSWORD_LENGTH

此参数控制新口令的最小长度。在不受信任的系统上，对根用户不可用。

MIN_PASSWORD_LENGTH=*N*

新口令必须包括至少 *N* 个字符。对于不受信任系统， *N* 可以为 6 到 8 之间的任意值。对于信任系统， *N* 可以为 6 到 80 之间的任意值。

缺省值： **MIN_PASSWORD_LENGTH=6**

NOLOGIN

此参数控制是否允许非根登录被 **/etc/nologin** 文件禁用。注意此参数仅应用于使用会话管理服务的应用程序，此服务由 *pam_hpsec*(5) 提供，在 **/etc/pam.conf** 中配置，或者其他间接调用的 *login*(1) ，如 *telnetd*(1M) 和 *rlogind*(1M) 命令。其他服务或者可能、或者不能选择强制使用 **/etc/nologin** 文件。

NOLOGIN=0

忽略 **/etc/nologin** 文件并在 **/etc/nologin** 文件存在时不退出。

NOLOGIN=1

显示 **/etc/nologin** 文件内容并在 **/etc/nologin** 文件存在时退出。

缺省值： **NOLOGIN=0**

NUMBER_OF_LOGINS_ALLOWED

此参数控制每个用户允许的同时登录次数。请注意仅对非根用户强制，并且仅应用于使用会话管理服务的应用程序，此服务由 *pam_hpsec*(5) 提供，在 **/etc/pam.conf** 中配置，或者其他间接调用的 *login*(1)，如 *telnetd*(1M) 和 *rlogind*(1M) 命令。

NUMBER_OF_LOGINS_ALLOWED=0

每个用户允许登录任意次数。

NUMBER_OF_LOGINS_ALLOWED=*N*

N 每个用户允许的登录次数。

缺省值： **NUMBER_OF_LOGINS_ALLOWED=0**

PASSWORD_HISTORY_DEPTH

此参数控制口令历史记录深度。对于特殊用户，新口令仅对照口令历史记录中最近使用的口令检查。用户不允许重新使用先前使用过的口令。

PASSWORD_HISTORY_DEPTH=*N*

对于特殊用户，新口令仅对照 *N* 最近使用的口令检查。

口令历史记录深度设置为 2 时，防止用户在两个口令之间替换。支持的最大口令历史记录深度为 10，支持的最小口令历史记录深度为 1。深度设置值大于 10 时当作 10 处理，深度设置值小于 1 时当作 1 处理。

口令历史记录深度配置基于系统，并仅由信任系统在文件数据库中对用户提供支持。此功能不支持 **NIS** 或 **NISPLUS** 数据库中的用户。一旦启用此功能，对系统中的所有用户进行相同的检查。如果此参数未配置，口令历史记录检查功能自动禁用。此功能禁用时，口令历史记录检查深度设置为 1。

口令更改服从其他所有新口令的规则，包括使用当前口令检查。

缺省值： **PASSWORD_HISTORY_DEPTH=1**

PASSWORD_MIN_<type>_CHARS

此格式的参数用于请求具有最小字符数的特殊类型新口令（大写、小写、数字或特殊字符）。此项功能在强制站点安全策略时有用，如选择不易被猜中的口令。

PASSWORD_MIN_UPPER_CASE_CHARS=*N*

指定在更改口令时需要至少 *N* 个大写字符。

PASSWORD_MIN_LOWER_CASE_CHARS=*N*

指定在更改口令时需要至少 *N* 个小写字符。

PASSWORD_MIN_DIGIT_CHARS=*N*

指定在更改口令时需要至少 *N* 个数字字符。

PASSWORD_MIN_SPECIAL_CHARS=*N*

指定在更改口令时需要至少 *N* 个特殊字符。

缺省值：这些参数的缺省值均为零。

PASSWORD_MAXDAYS

此参数控制口令有效期的最少天数缺省值。此值如果指定，在对给定用户不存在时限限制的口令更改验证过程中验证子系统。该值在口令更改后生效。此参数仅用于本地用户，不应用于信任系统。**passwd -x** 选项可用于覆盖特定用户的该项值。

PASSWORD_MAXDAYS=*N*

新口令最大有效期为 *N* 天，在那之后必须更改口令。

缺省值： **PASSWORD_MAXDAYS=-1** （关闭口令的时限）

PASSWORD_MINDAYS

此参数控制口令可以更改之前最少天数的缺省值。此值在对给定用户不存在时限限制的口令更改过程中验证子系统。该值长期存储并在口令更改后生效。此参数仅用于本地用户，不用于信任系统。**passwd -n** 选项可用于覆盖特定用户的该项值。

PASSWORD_MINDAYS=*N*

新的口令可以在上次更改之后，经过至少 *N* 天再修改。

缺省值： **PASSWORD_MINDAYS=0**

PASSWORD_WARNDAYS

此参数控制口令到期之前开始警告用户必须更改口令的指定天数。此值如果指定，在对给定用户不存在时限限制的口令更改验证过程中验证子系统。该值在口令更改后生效。此参数在映像口令系统中仅应用于本地户。 **passwd -w** 选项可用于覆盖特定用户的该项值。

PASSWORD_WARNDAYS=*N*

用户在口令过期前 *N* 天收到警告。

缺省值： **PASSWORD_WARNDAYS=0** （无警告）

SU_DEFAULT_PATH

此参数定义新的缺省 **PATH** 环境值，在 **su** 非超级用户帐户完成时进行设置。请参考 *su*(1) 。

SU_DEFAULT_PATH=*new_PATH*

在调用 **su** 命令时， **PATH** 环境变量设置为 *new_PATH* 。路径值未验证。此参数不显示给超级用户，并仅在 **su** 命令的“-”选项未使用时可用。

缺省值：如果此参数未定义或被加以注释，则 **PATH** 不更改。

SU_KEEP_ENV_VARS

此参数强制 **su** 传送某个“不安全”环境变量至其子进程，而不考虑这种做法的安全性风险。请参考 *su*(1) 。

su 的缺省情况下不导出环境变量 **HOME**、 **ENV**、 **IFS**、 **SHLIB_PATH** 或 **LD_*** ，这是因为它们会被错误用于不良用途。这些变量的任意组合可以在此条目中指定，变量使用逗号分隔。当前无其他环境变量可以此方式指定。在今后 HP-UX 版本中出于安全需求考虑可能会有所改变。

SU_KEEP_ENV_VARS=*var1***,***var2***,***...***,***varN*

缺省值：如果此参数未定义或被注释出，这些环境变量不会被 **su** 命令所传送。

SU_ROOT_GROUP

此参数定义 **su** 命令的根组名称。请参考 *su*(1) 。

SU_ROOT_GROUP=*group_name*

根组名称设置为指定的符号组名。 **su** 命令强制限制非超级用户必须为作为指定根组成员，该组被允许 **su** 至根。此项不改变口令检查。

缺省值：如果此值未定义或被注释出，则没有缺省值。这种情况下 允许非超级用户 **su** 至根，而无需绑定根组限制。

UMASK 此参数控制所有通过 *pam_unix*(5) 和（或） *pam_hpsec*(5). 开始会话的 *umask*(2) 。可用值为 0 到 0777 间的无符号八进制整数（前置零可以省略）。

UMASK=*default_umask*

umask 设置或进一步限制 *default_umask* 值。对于信任系统， **umask** 也受限制，以使其不超过 **/usr/include/hpsecurity.h** 中定义的 **SEC_DEFAULT_MODE** 。

缺省值： **UMASK=0**

注释

使用 *secdef*(3) 中定义的函数读取此文件中定义的参数值。

作者

security 文件由 HP 开发。

文件

/etc/default/security

另请参阅

login(1)、 passwd(1)、 su(1)、 init(1M)、 secdef(3)、 pam_hpsec(5)、 pam_unix(5)。

名称

services - 服务名称数据库

说明

文件 **/etc/services** 将正式的服务名和别名与服务使用的端口号和协议关联。对于每个服务，应当有一行带有如下信息：

<official service name> *<port number/protocol name>* *<aliases>*

端口号 0 到 1023 通过 RFC 1700 分配。此 RFC 也列出了编号大于 1023 的各端口号的常规用途。

别名是服务用于识别的其他名称。库例行程序（例如 **getservbyname()** ）可以使用服务别名来调用，而不是服务的正式名称。例如：

shell 514/tcp cmd

在此示例中， **getservbyname()** 可以使用 **cmd** 而不是 **shell** 来调用：

sp = getservbyname("cmd", "tcp");

而不是

sp = getservbyname("shell", "tcp");

都生成相同的结果。

行不能以空格或制表符开头 。项目通过任意数量的空白字符（空格或制表符的任意组合）来分隔。端口号和协议名称被认为是单个的 *item* 。 **/** 用于分隔端口和协议（例如， **512/tcp** ）。 **#** 字符指示注释的开始。搜索文件的例行程序不会解释从 **#** 开始直到行结束的字符。

服务名可能包含除空格、换行或注释字符之外的任意可打印字符。结尾的空白字符（空格或制表符）允许在行尾。

不是此文件中列出的所有服务都在 HP-UX 中可用。

举例

```
shell     514/tcp   cmd
telnet    23/tcp
login     513/tcp
```

作者

services 由加州大学伯克利分校开发。

文件

/etc/services

另请参阅

getservent(3N)。

名称

services.window - 包含应用程序和与其关联的内存窗口 ID 的文件

说明

文件 **/etc/services.window** 由应用程序利用内存窗口来使用。

/etc/services.window 文件中的每一行都将一个应用程序与一个内存窗口 ID 相关联。 **/etc/services.window** 文件中的行不能以一个空格或制表符开头。格式是一个定义该应用程序的唯一 *name* ，其后是一个空格/制表符，然后是一个唯一的 *window_id* 。请参阅举例中的示例文件。

内存窗口允许在一个唯一的或现有的内存窗口中启动一个进程，在此窗口中，它可创建对象并与同一内存窗口中的其他应用程序共享对象。

创建内存窗口，会删除系统在共享资源上的广泛限制。如果没有内存窗口，32 位进程会限制在 1.75 千兆字节的共享资源。每一个内存窗口都允许定义一个唯一的 1 千兆字节象限，而且由于在一个系统中可定义多个内存窗口，因此对于 32 位进程来说，系统的共享资源总数可超过 1.75 千兆字节。

定义内存窗口仅适用于 32 位进程。

/etc/services.window 文件为内存窗口应用程序提供一个中心空间，以便与它们的内存窗口 ID 相关联。在存在任何冲突的情况下，仅需要在 **/etc/services.window** 中进行更改以便为整个应用程序选择其他内存窗口。如果不使用 **/etc/services.window** ，并且用户应用程序的硬代码窗口 ID 位于它们的启动脚本中，则不容易检测或解决冲突。

一个内存窗口应用程序使用命令 *getmemwindow*(1M) 来从 **/etc/services.window** 文件中提取应用程序的 *window_id*，然后将该 ID 传送给 *setmemwindow*(1M) 。

使用相同的窗口 ID 将应用程序放置到同一内存窗口中。

举例

下面是一个 **/etc/services.window** 示例文件。

/etc/services.window

#

application1 20

application2 30

application3 40

作者

services.window 由 HP 开发。

文件

/etc/services.window 包含应用程序的关联窗口 ID 的文件。

另请参阅

setmemwindow(1M)、getmemwindow(1M)。

《11.0 Memory Window White Paper》

名称

shadow - 映像口令文件

概要

#include <shadow.h>

说明

/etc/shadow 文件通过 **pwconv** 命令由 **/etc/passwd** 文件创建。只有特权用户可读。它可以通过 **passwd**、**useradd**、 **usermod** 和 **userdel** 命令修改。程序可以使用在联机帮助页 *getspent*(3C) 中描述的接口访问该信息。这些函数返回一个指向 **spwd** 结构的指针，该指针在头文件 **<shadow.h>** 中定义。

字段

/etc/shadow 文件是一个包含任意数量用户条目的 ASCII 文件，这些条目由换行符分隔。每个用户条目行包含如下由冒号分隔的字段：

login name 每个 *login name* 必须匹配一个 **/etc/passwd** 中的登录名。 **pwconv** 按照 **/etc/passwd** 中条目的相同顺序将用户条目放入 **/etc/shadow** 中。

encrypted password
每个 **/etc/passwd** 条目的 *password* 字段都包含一个“x”，而实际加密口令位于 **/etc/shadow** 中。 *encrypted password* 字段由 13 个字符组成，这些字符是从 64 字符的“数字”字符集中选出的。这些用来代表“数字”的字符是 **.** 代表 0， **/** 代表 1， **0** 到 **9** 代表 2 到 11， **A** 到 **Z** 代表 12 到 37， **a** 到 **z** 代表 38 到 63。如果该字段为空，则没有口令，且在登录时不要求提供口令。如果输入的字符不是数字集中的部分（如 *），则禁止登录。

last change 从 1970 年 1 月 1 日到口令最后修改时间的天数。

min days 口令能够被更改前必须过期的最短时间，以天为单位。

max days 口令有效的最多天数。口令过期的用户试图登录时，将被强制提供一个新口令。如果 *min days* 和 *max days* 都是 0，那么用户将在下次登录时被强制更改口令。如果 *min days* 比 *max days* 大，则口令不能被更改。这些限制不适用于超级用户。

warn days 口令过期前提前警告用户的天数。

inactivity 口令过期后允许非活动状态的最大天数。口令过期后的指定天数内如果不更改口令，则帐户被锁定。如果该字段设置为 0，那么将要求用户更改其口令。

expiration 从 1970 年 1 月 1 日到帐户不再有效的绝对天数。这个字段的零值表示该帐户被锁定。

reserved *reserved* 字段始终为 0，并被保留以备将来使用。

注释

/etc/shadow 文件不适用于已转换为信任系统的系统。

文件

/etc/passwd 系统口令文件
/etc/shadow 映像口令文件

另请参阅

login(1)、 passwd(1)、 pwconv(1M)、 pwunconv(1M)、 useradd(1M)、 userdel(1M)、 usermod(1M)、 crypt(3C)、 getspent(3C)、 putspent(3C)、 nsswitch.conf(4)、 passwd(4)、 shadow(4)。

名称

shells - 允许的登录 Shell 列表

概要

/etc/shells

说明

/etc/shells 是一个 ASCII 文件，其中包含系统上合法的 Shell 的列表。每个 Shell 在文件中通过其绝对路径名列出。

假定以 **#** 开始的行或行的部分为注释并忽略。空白行也被忽略。

作者

shells 由 HP 和加州大学伯克利分校联合开发。

文件

/etc/shells

另请参阅

chsh(1)、 ftpd(1M)、 getusershell(3C)。

名称

slp.conf - SLP 代理的配置文件

概要

/etc/slp.conf

说明

/etc/slp.conf 文件包含不同的代理配置选项，其中包括使用 SLP API 客户端、服务代理服务器和目录代理。此文件确定主机上运行的所有 SLP 代理的配置。

SLP SA 服务器和 DA 在调用过程和通过发送 **SIGHUP** 信号进行重新配置过程中读取 **slp.conf** 文件。将配置文件中指定的属性读入进程内存。

其后的对此配置文件的修改不影响已经运行的代理操作，除非它们被重新配置或重新启动。

配置文件在被“UAs”调用时通过 SLP API 读取，用于确定配置参数的值，此参数在配置文件中指定， API 发送查询时会用到这些参数。

配置文件格式由零或更多属性定义的换行分隔列表组成。每个属性定义对应一个特定可配置的 SLP、网络或其他参数，这些参数在三个 SLP 代理的一个或多个中。ABNF [5] 语法中文件格式的语法为:

```
config-file   =  line-list
line-list     =  line / line line-list
line          =  property-line / comment-line
comment-line  =  ( "#" / ";" ) 1*allchar newline
property-line =  property newline
property      =  tag "=" value-list
tag           =  prop / prop "." tag
prop          =  1*tagchar
value-list    =  value / value "," value-list
value         =  int / bool /
                 "(" value-list ")" / string
int           =  1*DIGIT
bool          =  "true" / "false" / "TRUE" / "FALSE"
newline       =  CR / ( CRLF )
string        =  1*stringchar
tagchar       =  DIGIT / ALPHA / tother / escape
tother        =  %x21-%x2d / %x2f /
                 %x3a / %x3c-%x40 /
                 %x5b-%x60 / %7b-%7e
                 ; i.e., all characters except '.',
                 ; and '='.
stringchar    =  DIGIT / ALPHA / sother / escape
```

```
sother      =  %x21-%x29 / %x2a-%x2b /
               %x2d-%x2f / %x3a-%x40 /
               %x5b-%x60 / %7b-%7e
               ; i.e., all characters except ‘,’
allchar     =  DIGIT / ALPHA / HTAB / SP
escape      =  "
               ; Used for reserved characters
```

配置属性分解为下列类别:

DA 配置

静态范围配置

跟踪和日志

连续代理注册

网络配置参数

UA 配置

下列配置属性部分在 RFC2614 中指定，并被上述类别支持。

DA 配置

DA 配置属性在本节说明。

net.slp.isDA

如果将 SLP 服务器用作 DA 则指示布尔型。如果失败，运行 slpd 作为 SA 服务器。缺省值为假。

net.slp.DAHeartBeat

一个 32 位的整数给出 DA 心跳的秒数。缺省值为 3 小时（10800 秒）。该值在 isDA 为假时忽略。

net.slp.DAAttributes

逗号分隔的括号内属性和值的列表对， DA 必须在 DAAdverts 中进行通告。此属性必须在 SLP 属性列表线格式中，包括转义为保留字符。此属性目前被忽略。

静态范围配置

这些属性允许配置不同方面的范围处理。

net.slp.useScopes

字符串值列表，指示 UA 或 SA 在作出请求或注册时允许使用的范围，或者 DA 必须支持的范围。DA 和 SA 不存在、或者对 DHCP 没有可用的范围信息时，使用缺省范围“DEFAULT”。如果此参数没有在 UA 中使用或对 DHCP 没有可用的范围信息，使用用户范围模型。 UA 给出主动或被动的 DA 或 SA 发现所使用的所有范围可用信息。如果此信息在此处不能发现，则使用范围“DEFAULT”。与其他属性不同，此属性为“只读”。因而，任何在配置文件被读取之后试图更改此项的操作都将被忽略。

net.slp.DAAddresses

IP 地址或 DNS 解析的 SLPv2 DA 主机名值列表，在静态配置 UA 和 SA 发送请求和注册时必须使用。此值列表被 DA 忽略（除非 DA 同时也是 SA 服务器）。缺省值尚无。与其他属性不同，此属性为“只读”。因而，在配置文件被读取之后任何试图更改此项的操作都将被忽略。

下列语法描述了属性:

```
addr-list   = addr / addr "," addr-list
addr        = fqdn / hostnumber
fqdn        = ALPHA / ALPHA *[ anum / "-" ] anum
anum        = ALPHA / DIGIT
hostnumber  = 1*3DIGIT 3("." 1*3DIGIT)
```

示例:

sawah,mandi,sambal

DNS 没有配置时， IP 地址可用于替代网络中的主机名，需要提醒网络管理员的是使用 IP 地址将导致机器重新编号复杂化，因为必须更改静态配置网络的 SLP 配置属性文件。类似的，如果使用主机名，实现程序必须小心名称服务在 SLP 开始之前可用。换而言之， SLP 不能用于查找名称服务。

跟踪和日志

本节说明使用不同代理时打印出的跟踪和日志信息。

net.slp.traceDATraffic

布尔型控制，打印 DA 流消息。缺省值为假。

net.slp.traceMsg

布尔型控制，打印 SLP 详细消息。打印所有输入消息和输出回应中的字段。缺省值为假。

net.slp.traceDrop

布尔型控制，在 SLP 消息断开时打印详细消息。缺省值为假。

net.slp.traceReg

布尔型控制，转储全部注册服务，包括注册和取消注册。如果为真，DA 和 SA 服务器的内容在发生注册或取消注册后转储。缺省值为假。

网络配置属性

本节的属性允许设置不同网络属性。

net.slp.isBroadcastOnly

布尔型，指示是否使用广播代替组播。一般无需设置此项，因为 SLP 将在组播不可用时自动使用广播。缺省值为假。

net.slp.passiveDADetection

布尔型，指示是否使用被动 DA 检测。缺省值为真。

net.slp.multicastTTL

小于或等于 255 的正整数，给出组播的 TTL。缺省值为 255。

net.slp.DAActiveDiscoveryInterval

16 位正整数，给出 DA 有效发现请求间隔的秒数。缺省值为 900 秒（15 分钟）。如果该属性设置为零，有效发现关闭。在可用的 DA 从 **DHCP** 或 **net.slp.DAAddresses** 中获得的属性严格限制时有用。

net.slp.multicastMaximumWait

32 位整数，给出执行组播的最大时间，以毫秒为单位。缺省值为 15000 ms（15 秒）。

net.slp.multicastTimeouts

当前未使用此参数值。当前组播的超时在内部根据 net.slp.multicastMaximumWait 参数生成。

net.slp.DADiscoveryTimeouts

当前未使用此参数值。当前组播的超时在内部根据 net.slp.multicastMaximumWait 参数生成。

net.slp.datagramTimeouts

当前未使用此参数值。当前组播的超时在内部根据 net.slp.unicastMaximumWait 参数生成。

net.slp.randomWaitBound

32 位整数给出全部随机等待参数的最大值，以毫秒为单位。缺省值为 1000 ms（1 秒）。

net.slp.MTU

16 位整数，给出网络数据包 MTU，以字节为单位。此项为待发送的数据报大小的最大值，但在实现中可能收到更大的数据报。最大大小包括 IP 和 UDP 或者 TCP 头。缺省值为 1400 字节。

net.slp.interfaces

字符串值列表给出网络接口 IP 地址，此处 DA 或 SA 监听端口 427 的组播、单播 UDP 和 TCP 消息。缺省值为使用全部网络接口。

示例：

195.42.42.42,195.42.142.1,195.42.120.1

示例机器有三个 DA 应该监听的接口。

请注意由于此属性仅使用 IP 地址，在网络重编号后需要更改。

SA 配置

本节说明 SA 的配置属性。这些属性通常由 SA 程序设置，这是因为它们指定给每个 SA。

net.slp.SAAttributes

逗号分隔的括号内属性和值的列表对， SA 必须在 SAAdverts 中进行通告。此属性必须在 SLP 属性列表导线格式中，包括转义为保留字符。

当前忽略此属性。

UA 配置

本节说明 UA 的配置属性。这些属性可以通过 UA 程序或配置文件设置。

net.slp.locale

语言环境的 RFC 1766 语言标记。设置此属性将使得 SLP 消息的属性值成为缺省语言环境。缺省值为“en”。此属性也用于 SA 和 DA 配置。

当前仅能识别“en”（英语）语言，忽略其他语言标记。

net.slp.maxResults

32 位整数，给出超时前同步请求的积累和返回结果的最大数，或者在请求结果报告为异步时回调返回结果的最大数。

DA 和 SA 始终返回与请求匹配的全部结果。此配置值仅应用于 UA，过滤器输入结果并仅返回 net.slp.maxResults 指示相同的结果数。

net.slp.typeHint

服务类型名称的值列表。没有 DA 时， UA 执行 SA 发现以查找范围。这些 SA 发现请求可能包括请求服务类型作为属性。

API 实现将使用此属性的服务类型名称以发现支持所需服务类型的 SA （及其范围）。例如，如果 net.slp.typeHint 设置为“service:imap,service:pop3”，那么 SA 发现请求将包括搜索过滤器:

(|(service-type=service:imap)(service-type=service:pop3))

API 库也可使用单播联系已发现的 SA，随即请求这些服务类型，以优化网络访问。

net.slp.securityEnabled

指示是否所有代理应使用验证块。当前忽略。

作者

slp.reg 由 Caldera Systems, Inc. 开发。

另请参阅

slpd(1M)、slp.reg(4)。

RFC 2614、RFC 2608。

名称

slp.reg - SLP 静态注册文件

概要

/etc/slp.reg

说明

/etc/slp.reg 文件为不启用 SLP 并且不能转换的老应用，并为 SLP 实现之间的活动交换注册提供了一种机制。

该文件包含一个服务注册列表，这些注册由 slpd 在启动时读取并在之后由 slpd 公布，而 slqd 既可以作为服务代理服务器也可以作为目录代理。

注册的字符编码需要是 UTF-8。

以 ABNF 格式 [5] 的串行注册文件的语法如下：

```
ser-file      =  reg-list
reg-list      =  reg / reg reg-list
reg           =  creg / ser-reg
creg          =  comment-line ser-reg
comment-line  =  ( "#" / ";" ) 1*allchar newline
ser-reg       =  url-props [slist] [attr-list] newline
url-props     =  surl "," lang "," ltime [ "," type ] newline
surl          =  ;The registration's URL. See
                 ; [8] for syntax.
lang          =  1*8ALPHA [ "-" 1*8ALPHA ]
                 ;RFC 1766 Language Tag see [6].
ltime         =  1*5DIGIT
                 ; A positive 16-bit integer
                 ; giving the lifetime
                 ; of the registration.
type          =  ; The service type name, see [7]
                 ; and [8] for syntax.
slist         =  "scopes" "=" scope-list newline
scope-list    =  scope-name / scope-name "," scope-list
scope         =  ; See grammar of [7] for
                 ; scope-name syntax.
attr-list     =  attr-def / attr-def attr-list
attr-def      =  ( attr / keyword ) newline
keyword       =  attr-id
attr          =  attr-id "=" attr-val-list
attr-id       =  ;Attribute id, see [7] for syntax.
attr-val-list =  attr-val / attr-val "," attr-val-list
```

```
attr-val     = ;Attribute value, see [7] for syntax.
allchar      = char / WSP
char         = DIGIT / ALPHA / other
other        = %x21-%x2f / %x3a-%x40 /
               %x5b-%x60 / %7b-%7e
               ; All printable, nonwhitespace US-ASCII
               ; characters.
newline      = CR / ( CRLF )
```

访问名称、属性标记和属性值的语法需要对特殊字符进行转义。处理串行注册的 DA 和 SA 服务器必须完全按照象 SA 注册的那样处理它们。在 url-props 产品中，类型标记是可选的。如果对 service:URL 存在类型标记，则表示一个警告并忽略类型名称。如果指定了最大周期 **ltime** （65535 秒），则该注册作为永久注册，并被 DA 或 SA 服务器不断刷新直到它退出。

范围可以包含在注册中，只需在 url-props 产品后面用标记“scope”包含一个属性定义，而标记“scope”后面是以逗号分隔的范围名称列表。如果有可选的范围，则注册在指定范围中生成。否则，它们在 DA 或 SA 服务器通过 net.slp.useScopes 属性配置的范围内进行注册。

如果范围列表包含不在 net.slp.useScopes 属性（已设置的属性提供）中的范围，或者不是由 DHCP 指定，则 API 库应拒绝注册并发出警告消息。

作者

slp.reg 由 Caldera Systems, Inc. 开发。

另请参阅

slpd(1M)、libslp(3N)、slp.conf(4)、slp_syntax(7)。

RFC 2614、RFC 2608。

名称

sm、 sm.bak、 state - statd 目录和文件结构

概要

/var/statmon/sm

/var/statmon/sm.bak

/var/statmon/state

说明

/var/statmon/sm 和 **/var/statmon/sm.bak** 是由 **statd** 生成的目录（请参阅 *statd*(1M)）。 **/var/statmon/sm** 中的每个文件表示有一台或多台计算机在其恢复时要由 **statd** 守护程序来通知。 **/var/statmon/sm.bak** 中的每个文件表示一个或多个由 **statd** 守护程序在其恢复时通知的计算机。

/var/statmon/state 是由 **statd** 生成的文件，以记录其版本号。版本号在每次崩溃或恢复时递增。

另请参阅

lockd(1M)、 statd(1M)。

名称

snmpd.conf - SNMP 代理的配置文件

说明

调用时，SNMP 代理从 **/etc/SnmpAgent.d/snmpd.conf** 配置文件读取配置信息。SNMP 代理为 *snmpd*(1M) （包括在 HP-UX 中）或者 **snmpd.ea** （需购买 HP OpenView 产品）。如果 **/etc/SnmpAgent.d/snmpd.conf** 中没有设置值，即 **/etc/SnmpAgent.d/snmpd.conf** 为空时，SNMP 代理无法回复。

参数

/etc/SnmpAgent.d/snmpd.conf 文件包含以下可配置值：

get-community-name
指定代理的团体名称。代理使用团体名称响应 SNMP 的 GetRequests。可以配置代理以响应多个获取团体名称。如果团体名称没有键入，代理 不响应 SNMP GetRequests。

set-community-name
为代理指定团体名称。代理使用此团体名称响应 SNMP SetRequests 和 SNMP GetRequests。可以配置代理以响应多个设置团体名称。如果设置团体名称没有键入，代理不响应 SetRequests。

trap-dest 指定系统往何处发送陷阱（也就是，陷阱目标）。此系统通常为管理器的 IP 地址。如果陷阱需要发送到多个系统，则每个系统中均需包括 *trap-dest* 行。

location 指定代理的物理位置。

contact 指定此代理的用户响应和如何联系该用户的信息。

SNMPv3 功能

使用 SNMP 的 v3 功能语法如下所示。格式如下：

TAG VALUE

其中 *TAG* 为下列之一：

usmUserEntry **usmUserEntry** 用于配置 SNMPv3 用户。

vacmSecurityToGroupEntry
vacmSecurityToGroupEntry 用于为组分配 *principal* ， *principal* 可以为 SNMPv3 用户或 SNMPv1、SNMPv2 组字符串。

vacmViewTreeFamilyEntry
vacmAccessEntry 用于定义组和关联的访问权限。

snmpTargetAddrEntry
snmpTargetAddrEntry 用于配置目标地址（通知发送到的地方）。

snmpNotifyEntry **snmpNotifyEntry** 用于配置通知条目。

snmpTargetParamsEntry

snmpTargetParamsEntry 用于配置发送通知时使用的参数。

VALUE 对任何给定 *TAG* 为有效值。当 *TAG* 为 **usmUserEntry** 时， *VALUE* 子句格式如下所示：

usmUserEngineID usmUserName usmUserAuthProtocol usmUserPrivProtocol \
usmUserStorageType usmTargetTag

其中：

usmUserEngineID 为八进制字符串，是授权 SNMP 引擎的管理的唯一标识符。对于 **snmpget/snmpset** 请求，代理配置文件的值为 *localSNMPID* 。

usmUserName 为 ASCII 文本的用户名。

usmUserAuthProtocol 是验证协议，用于代表 SNMP 引擎发送和接收消息。当前支持值为 *usmNoAuthProtocol* 和 *usmHMACMD5AuthProtocol* 。

usmUserPrivProtocol 为私密性协议，用于代表 SNMP 引擎发送和接收消息。当前不支持协议。

usmUserPrivProtocol 的缺省值为 OID，即 **.1.3.6.1.6.3.10.1.2.1** 。

usmUserStorageType 是 **nonVolatile** 、 **permanent** 或 **readOnly** 。

usmTargetTag 为 ASCII 文本，用于源地址检查。用于从 **snmpTargetAddrTable** 中选择一组条目。如果不需要源地址检查，该值为“-”。

当 *TAG* 为 **vacmSecurityToGroupEntry** 时， *VALUE* 子句格式如下所示：

vacmSecurityModel vacmSecurityName vacmGroupName vacmSecurityToGroupStorageType

其中：

vacmSecurityModel SNMPv1 中为 **snmpv1** ，SNMPv2c 中为 **snmpv2c** 和 SNMPv3 中为 **usm** 。

vacmSecurityName 为 ASCII “principal”字符串（SNMPv3 用户或者 SNMPv1/SNMPv2 组字符串）。

vacmGroupName 为 ASCII 文本定义的组名称。此组名称必须由至少一个 **vacmAccessEntry** 定义。

vacmSecurityToGroupStorageType

是 **nonVolatile** 、 **permanent** 或 **readOnly** 。

当 *TAG* 为 **vacmAccessEntry** 时， *VALUE* 子句格式如下所示：

vacmGroupName vacmAccessContextPrefix vacmAccessSecurityModel
vacmAccessSecurityLevel vacmAccessContextMatch vacmAccessReadViewName
vacmAccessWriteViewName vacmAccessNotifyViewName vacmAccessStorageType

其中：

vacmGroupName 为 ASCII 文本，代表组名称。

vacmAccessContextPrefix 为 ASCII 字符串，用于在管理请求中部分或完全匹配内容名称。短线“-”代表缺省内容。

vacmAccessSecurityModel SNMPv1 中为 **snmpv1** ，SNMPv2c 中为 **snmpv2c** 和 SNMPv3 中为 **usm** 。

vacmAccessSecurityLevel 为验证和私密的等级。当前支持的值在无验证无私密时为 **noAuthNoPriv** ，在有验证无私密时为 **authNoPriv** 。

vacmAccessContextMatch 为 **exact** 或 **prefix** ，用于指示请求内容如何必须匹配 *vacmAccessContextPrefix* 。

vacmAccessReadViewName 用于定义查看子树以获取请求。必须被至少一个 **vacmViewTreeFamilyEntry** 定义。

vacmAccessWriteViewName 用于定义查看子树以设置请求。必须被至少一个 **vacmViewTreeFamilyEntry** 定义。

vacmAccessNotifyViewName 用于定义查看子树，从中对象可以作为 VarBinds 包含在 Trap 消息和 Inform 请求中。必须被至少一个 **vacmViewTreeFamilyEntry** 定义。

vacmAccessStorageType 是 **nonVolatile** 、 **permanent** 或 **readOnly** 。

当 *TAG* 是 **vacmViewTreeFamilyEntry** 时， *VALUE* 子句格式如下所示：

vacmViewTreeFamilyViewName vacmViewTreeFamilySubtree
vacmViewTreeFamilyMask vacmViewTreeFamilyType
vacmViewTreeFamilyStorageType

其中：

vacmViewTreeFamilyViewName
是此子树查看的一系列名称。

vacmViewTreeFamilySubtree 是定义子树的对象标识符。

vacmViewTreeFamilyMask 是在 0x00 和 0xff 之间的十六进制数序列，用于限制 *vacmViewTreeFamilySubtree* 的值。0 值指示“wild card”（与所有匹配），1 值 指示精确匹配。

vacmViewTreeFamilyType 为 **included** 或 **excluded** ，表明 OID 下 *vacmViewTreeFamilySubtree* 定义的子树是否可以访问。

vacmViewTreeFamilyStorageType
是 **nonVolatile** 、 **permanent** 或 **readOnly** 。

当 *TAG* 为 **snmpTargetAddrEntry** 时， *VALUE* 子句格式如下：

snmpTargetAddrName snmpTargetAddrTDomain snmpTargetAddrTAddress
snmpTargetAddrTimeout snmpTargetAddrRetryCount snmpTargetAddrTagList

snmpTargetAddrParams snmpTargetAddrStorageType snmpTargetAddrTMask snmpTargetAddrMMS

其中:

snmpTargetAddrName 为 ASCII 文本，代表目标名称。

snmpTargetAddrTDomain 为指示网络类型的 OID。当前支持值为 **snmpUDPDomain** ，也就是 **1.3.6.1.6.1.1** 。

snmpTargetAddrTAddress 为 *x.x.x.x*:*y* ， *x.x.x.x* 是有效 IP 地址， *y* 是有效 UDP 端口号。地址用作外发通知的目的地地址。如果 *y* 为 0， **SR_TRAP_TEST_PORT** 的值用作目标端口号。否则，如果设置了 **SR_SNMP_TEST_PORT** ，则目的端口比 **SR_SNMP_TEST_PORT** 大 1，否则目的端口为 162。

snmpTargetAddrTimeout 用于 Inform 请求，估算往返时间（以百分之一秒为单位）。当 Inform 请求发送到该地址，而响应尚未到达时，SNMP 条目将假定响应不被发送。

按照 RFC-2573,缺省值为 1500（15 秒）。

snmpTargetAddrRetryCount 为时间数，如果没有收到响应，重新发送 Inform 请求。

RFC-2573 建议的缺省值为 3。

snmpTargetAddrTagList 为引用字符串，根据 **snmpNotifyTable** 中的 *snmpNotifyTag* 的值相应包含一个或多个标记。在 *snmpNotifyTable* 中定义的通知将发送给 *snmpTargetAddrTDomain* ，如果通知的 *snmpNotifyTag* 出现在此标记列表中。

snmpTargetAddrParams 为 ASCII 字符串，用于选择 **snmpTargetParamsTable** 中的值。

snmpTargetAddrStorageType 是 **nonVolatile** 、 **permanent** 或 **readOnly** 。

snmpTargetAddrTMask 为 *snmpTargetAddrTAddress* 的掩码值。

snmpTargetAddrMMS 是本地主机间可以传送的最大消息大小，主机地址 *snmpTargetAddrTAddress* ，无记录块。

缺省大小为 2048。

当 *TAG* 为 **snmpNotifyEntry** 时， *VALUE* 子句格式如下所示:

snmpNotifyName snmpNotifyTag snmpNotifyType snmpNotifyStorageType

其中:

snmpNotifyName 为 ASCII 文本，代表通知名称。

snmpNotifyTag 为 ASCII 字符串，用于选择 **snmpTargetAddrTable** 中的条目。

snmpNotifyType “1”代表陷阱，“2”代表通知。

snmpNotifyStorageType 是 **nonVolatile** 、 **permanent** 或 **readOnly** 。

当 *TAG* 为 **snmpTargetParamsEntry** 时， *VALUE* 子句格式如下：

snmpTargetParamsName snmpTargetParamsMPModel
snmpTargetParamsSecurityModel snmpTargetParamsSecurityName
snmpTargetParamsSecurityLevel snmpTargetParamsStorageType

其中：

snmpTargetParamsName 是 ASCII 文本，代表参数名称。

snmpTargetParamsMPModel SNMPv1 中为 **0** ， SNMPv2c 中为 **1** ， SNMPv3 中为 **3** 。此字段与 *snmpTargetParamsSecurityModel* 组合，定义待发送的通知类型。

snmpTargetParamsSecurityModel
SNMPv1 中为 **snmpv1** ， SNMPv2c 中为 **snmpv2c** ， SNMPv2* 中为 **snmpv2s** ，或SNMPv3 中为 **usm** 。此字段与 *snmpTargetParamsMPModel* 组合，定义待发送的通知类型。

snmpTargetParamsSecurityName
为 ASCII“principal”字符串（SNMPv3 用户或者 SNMPv1/SNMPv2 组字符串），用作通知。

snmpTargetParamsSecurityLevel
为待发送通知的安全级别。仅支持 **noAuthNoPriv** 值。

snmpTargetParamsStorageType
是 **nonVolatile** 、 **permanent** 或 **readOnly** 。

举例

用空格或制表位分隔字段。 **#** 字符指示注释的开头，忽略从 **#** 字符到行结尾的全部字符。

下列 **snmpd.conf** 文件示例中的每一行之前都有注释（以 **#** 开头），用于解释条目。

```
# Restrict the agent to responding only to
# SNMP GetRequests  that have the
# community name "secret"
get-community-name:  secret
# Allow the agent to respond to SNMP Get  and SetRequests with
# either the community name "private"  or "secret"
set-community-name: private
set-community-name:  secret
# Allow the agent to respond to SNMP Get  and SetRequests
# that have the community name "private"
set-community-name:  private
```

```
# Send traps to system 15.2.113.233
trap-dest:  15.2.113.233
# Specify the agent is located on  the first floor
# near the mens room
location:  1st Floor near Mens Room
# Specify Bob Jones  is responsible for this agent
# and his phone number is  555-2000
contact: Bob Jones (Phone 555-2000)

# Create a SNMPv3 user 'v3usr' with No Authentication Protocol.
usmUserEntry  localSnmpID v3usr usmNoAuthProtocol 1.3.6.1.6.3.10.1.2.1 \
nonVolatile  whereValidRequestsOriginate -
# Create a SNMPv3  user 'v3usr' with Authentication enabled and
# password  as "passwd".
usmUserEntry localSnmpID v3usr usmHMACMD5AuthProtocol  1.3.6.1.6.3.10.1.2.1 \
nonVolatile whereValidRequestsOriginate  "passwd"
# Create a group 'admin' and make the  user 'v3usr' a part of the
# same group.
vacmSecurityToGroupEntry  usm v3usr admin nonVolatile
# Assign access control the group 'admin'. This group will have
# security protocol as no authentication and no privacy
vacmAccessEntry admin  - usm noAuthNoPriv prefix All All - nonVolatile
#  'All' is the name of the view that will define the access for the
#  group 'admin'. Give access to the view named 'All'. The access is
#  for the subtree 'internet' i.e. .1.3.6.1
vacmViewTreeFamilyEntry   All 1.3.6.1 - included nonVolatile
# Create  a target address entry for 192.168.40.40 with UDP port as 0.
#  If SNMP_TRAP_TEST_ENTRY or SNMP_TEST_PORT_ENTRY are not used,
#  default value of UDP port 162 will be used.
snmpTargetAddrEntry  stae2 1.3.6.1.6.1.1 192.168.40.40:0 0 0 \
whereValidRequestsOriginate  - nonVolatile 255.255.255.255:0 2048
```

作者

snmpd.conf 由 HP 开发。

文件

"HP-UX 11.X 和 Solaris 2.X" **/etc/SnmpAgent.d/snmpd.conf**

另请参阅

snmpd(1M)、 snmpd.ea(1M)、 RFC 1155、RFC 1157、RFC 1212、2RFC 1213、RFC 1231、RFC 1398。

名称

softkeys - keysh 功能键文件格式

背景

keysh 功能键信息以功能键节点层次结构的格式存储。此层次结构的最高级代表功能键命令自身；较低级代表不同命令选项和参数。

功能键标签将 窗口组合为此功能键的节点层次结构，用户可通过它查看和选择 合格的节点。满足下列条件时，节点合格：

- 在缺省情况下处于启用状态，后来没有被某些同级节点禁用，或
- 在缺省情况下处于禁用状态，后来没有被某些同级节点禁用，但被某些同级节点启用。

选择某个功能键节点时，该节点相应可启用或禁用任意其同级节点。进入功能键节点层次结构的新窗口计算如下：

- 如果选定的节点不是叶节点，显示其合格子节点；
- 否则，如果该节点仍保留有合格同级节点，则重新显示；
- 另外，如果该节点的父节点仍有合格的同级节点，将重新显示这些节点等内容，并上移节点层次结构。

这种节点显示和选择的过程将继续，直至用户输入了完整命令。

在该情况下， **keysh** 执行与每个选定功能键节点关联的 编辑规则。这些编辑规则创建供 Shell 执行的 HP-UX 命令。

功能键文件格式

每个功能键文件包括一个或多个功能键定义，并表现为 “功能键节点”的子层次结构。

功能键节点有两种基本类型：

option “选项”显示在功能键标签上并在选中后向命令行中插入文字文本。示例为命令和选项名称。

string “字符串”（或“参数”）在功能键标签上显示，但在选中后不向命令行中插入文本；而是显示提示消息。用户必须向命令行中输入所需文本。示例为文件和用户名。

请注意关键字 **softkey** 可用作关键字 **option** 的同义名称。

基本功能键节点定义由下列部分组成：

{**option** | **string**} *softkey*
attribute
.
.
.

;

softkey 为功能键节点名称，由此派生出命令行文本和功能键标签。如果需要， *softkey* 中一个单独的加号 **(+)** 可用于在音节边界强制连接功能键标签。

如果功能键节点有相关的子菜单，其末尾的 **;** 将按如下方式替换为子节点列表：

{
softkey node
.
.
.
}

每个功能键节点可以有下列可选 *attribute* 字段：

disable *count*	选择此节点将禁用此功能键右侧的 *count* 功能键节点，- 缺省值为 0。
enable *count*	选择此功能键将启用此功能键右侧的 *count* 功能键节点，- 缺省值为 0。
{**filter**|**command**}	此节点仅分别对过滤器或命令有效，- 缺省值为两者之一。
{**motorola**|**precision**}	此节点仅分别在 **keysh** 使用 Motorola (MC680x0) 或 precision (PA-RISC) 处理器运行时有效，- 缺省值为二者之一。
disabled	此节点开始时为禁用，必须在启用后使用，- 缺省值为开始时启用。
automatic	选中此节点时自动输入命令。
editrule *editrule*	此节点的编辑规则。
cleanuprule *editrule*	与此功能键命令关联的所有其他编辑规则 *after* 执行的编辑规则 - 每个功能键命令只允许一个清理规则。
hint *string*	该节点的一行提示 - 仅对“string”功能键节点有效。
help *helptext*	此节点的帮助信息（可能不止一行）。
required *string*	如果未选中此节点，显示一行错误消息。

参数如下：

count	有符号的整数，为单词 **none** 或单词 **all** 。
editrule	编辑规则（有关说明参阅下文）。
helptext	用引号括起的 **nroff** 样式的帮助信息（有关说明另请参阅下文）。
string	包含在引号中的任意字符串。请注意引号中， **** 在使用 *awk*(1) 时转义下一个字符。

典型备份功能键节点定义类似于：

backup softkey *softkey* [**literal** *literal*] **;**

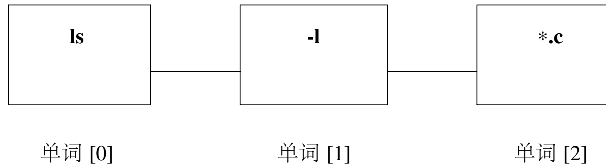
此处 *literal* 为文字文本字符串，用于编程终端功能键（如果与 *softkey* 不同）。

功能键文件中未用引号括起的散列标记字符 **(#)**，将其后到行尾的内容界定为注释。

功能键命令 转换

要将功能键命令转换为 HP-UX 命令， **keysh** 执行与用户选定的功能键相关联的 *editrules* 。这些编辑规则通过 **awk**-like 编辑语言创建 “单词列表”。这些转换单词列表传递给 Shell 用于执行。

对于简单转换，此列表类似于：



单词 [0] 单词 [1] 单词 [2]

每次调用编辑规则，特殊常量 **last** 和 **next** 分别定义为列表中最后一个单词的索引（此示例中为“2”）和列表中可能的下一个单词（此示例中为“3”）。此外，常量 **argument** 设置为与用户为功能键的输入相同（例如，该例中功能键的 ***.c** 与文件名相对应）。

请注意 **keysh** 自动来回转换数字和字符串，此计算在执行编辑规则必需。此外，变量只在第一个编辑规则与功能键命令关联之前清除。所有分配变量可为后续编辑规则使用。

编辑规则

编辑规则是大括号内的编辑语句的列表（即 **{** 和 **}** ）。

编辑语句是：

- 其后为 **;** 的表达式
- **if** 语句，或者
- 单词分配语句。

表达式

简单表达式可为下列之一：

variable	从 a 到 z 的单字母
number	无符号整数
string	在引号中
char	在引号中
last	见上
next	见上
argument	见上

motorola	布尔型标志
precision	布尔型标志
command	布尔型标志
filter	布尔型标志
word[*number* **]**	见上

简单表达式可以与下列任意组合:

string[*number*]	单字符子字符串
string[*number*,*number*]	多字符子字符串
number+*number*	加号
number-*number*	减号
*number***number*	乘号
number/*number*	除号
number%*number*	模数
string&*string*	连接号
-*number*	非
string==*string*	等
string!=*string*	不等
number>=*number*	大于或等于
number<=*number*	小于或等于
number>*number*	大于
number<*number*	小于
number&&*number*	逻辑与
number||*number*	逻辑或
!*number*	逻辑否
(*string*)	组

还支持下列函数并返回指定结果:

strlen(*string*)	*string* 中字符数
strchr(*string*,*char*)	*string* 中的第一个 *char* 索引，或者为 −1
strrchr(*string*,*char*)	*string* 中最后一个 *char* 索引，或者为 −1
trim(*string*)	没有前置或结尾空格的 *string*
hex(*number*)	*number* 十六进制数，有前置 **0x**
octal(*number*)	*number* 八进制数，有前置 **0**

使用下列任意之一进行分配:

variable=*string*	简单分配

variable+=*number*	添加并分配
variable-=*number*	减去并分配
*variable**=*number*	乘以并分配
variable/=*number*	除以并分配
variable%=*number*	模数并分配
variable&=*string*	连接并分配
word[*number***]=***string*	简单分配
word[*number***]+=***number*	添加并分配
word[*number***]-=***number*	减去并分配
word[*number***]*=***number*	乘以并分配
word[*number***]/=***number*	除以并分配
word[*number***]%=***number*	模数并分配
word[*number***1]&=***string*	连接并分配

if 语句

if 语句与 awk 中整块模式 **if** 语句相同，结构如下：

```
if(number) {
    edit statement
    .
    .
    .
} else {
    edit statement
    .
    .
    .
}
```

else 部分可选。如果 *number* 非零，执行 *edit statement* 第一块。否则，如果存在第二个 *edit statement* 块，执行该块。

单词分配语句

单词分配语句包括下列：

insert(*number*，*string***);**	在 **word[***number***]**. 之前向单词列表中插入 *string* 作为新单词。
append(*string***);**	立刻在单词列表中的最后一个单词之后向单词列表中插入 *string* 作为新单词。等效于 **insert(next,** *string***);**.
dash(*string***);**	追加 *string* 到单词列表中的最后一个单词， *if* 该单词以短线开头。否则，立刻在单词列表的最后一个单词之后向单词列表中插入短线作为新单词，在这之后追加 *string* 。

delete(*number***);**	从单词列表中删除 **word[***number***]** 。

帮助文本

每个功能键节点可以有一个关联的 帮助文本，在用户需要帮助时显示。此帮助文件为闲置格式，通过首选寻呼显示给用户。

帮助文本格式类似与 nroff 语言，支持 *man*(5) 宏子集，用于写入标准 HP-UX 手册条目。尤其是此子集包括：

.nf	开始 **no-fill** 模式。显示文本 as-is，保留 new-lines 和 spaces，直至 **.fi**。
.fi	恢复 填充模式。显示填充到每个输出行的单词文本，尝试使用 90% 的屏幕宽度。（这是缺省模式）。
.br	在当前输出行中强制 中断。显示下一行的后续文本。
.sp	强制中断，然后显示一个空白行（垂直 空格）。
.P	强制中断，显示一个空白行，然后以无缩进开始新 段落。
.IP *tag indent*	强制中断，显示一个空白行，然后显示指定 *tag* ，接着使用指定 *indent* 开始新的 缩进段落。
.IL *tag indent*	开始新的 缩进行（与 **.IP** 相同，除了没有显示空白行）。

请注意这些宏在输入帮助文本中被识别为 任意位置，不仅仅在行的开始。同样所有宏参数 必须存在，即使只由引用的空白字符串组成。

举例

对于自定义 **cd** 命令（请参阅 *cd*(1) ）：

```
softkey cd
editrule { append("cd"); }
{
 softkey keysh-src disable all
 editrule { append("~/keysh/src"); }
 ;
 softkey keysh-test disable all
 editrule { append("~/keysh/test"); }
 ;
 softkey keysh-doc disable all
 editrule { append("~/keysh/doc"); }
 ;
 softkey demo disable all
 editrule { append("~/demo"); }
 ;
 softkey tmp disable all
```

```
   editrule { append("/tmp"); }
   ;
   string <dir> disable all
   editrule { append(argument); }
   required "Enter the name of the directory to move to."
   ;
  }
```

有关其他示例的信息，请参阅文件 **/usr/lib/keysh/C/softkeys** 。

作者

keysh 由 HP 和 AT&T 联合开发。

文件

$HOME/.softkeys	用户功能键定义文件
/usr/lib/keysh/$LANG/softkeys	标准功能键定义文件

另请参阅

keysh(1)、man(5)。

名称

symlink - 符号链接

说明

符号（或 软）链接是一个名称间接参考（指向）相对或绝对路径名的文件。

路径名解释期间，到相对路径名的符号链接扩展到被解释的路径名，而到绝对路径名的符号链接替换为被解释的路径名。

因此，对于路径名 **/a/b/c/d** ：

如果 **c** 是一个到相对路径名的符号链接如 **../x/y** ，则路径名被解释为 **/a/b/../x/y/d** 。

如果 **c** 是一个到绝对路径名的符号链接如 **/v/w** ，则路径名被替换为 **/v/w/d** 。

所有符号链接都以这种方式解释，只有一个例外：当符号链接是路径名的最后一个组成部分时，它被作为以下系统调用之一的参数进行传递： **readlink** 、 **rename** 、 **symlink** 、 **unlink** 、 **chown** 或 **lstat** （请参阅 *readlink*(2)、*rename*(2) 、 *symlink*(2) 、 *unlink*(2) 、 *chown*(2) 和 *lstat*(2) ）。对这些调用，符号链接自身被访问或受到影响。

与普通（硬）链接不同，符号链接可以指向任意路径名并可以跨不同的逻辑设备（卷）。

路径名可以是任意类型文件的（包括目录或另一个符号链接），如果系统中不存在那样的路径则可能无效。（可能通过将符号链接指向它们自身或其他符号链接来使它们形成一个闭环。系统通过限制翻译路径名时遍历的符号链接数来检测这种情况）。

符号链接的模式和所有权被系统忽略，表示 **chmod** 影响实际的文件，而不是包含该符号链接的文件（请参阅 *chmod*(1) ）。

符号链接可以使用 **ln** 或 **symlink** 创建（请参阅 *ln*(1) 和 *symlink*(2) ）。

作者

symlink 由 HP 和加州大学伯克利分校联合开发。

另请参阅

cp(1)、symlink(2)、readlink(2)、link(2)、stat(2)、mknod(1M)。

名称

system - 系统描述配置文件

说明

HP-UX 系统描述文件描述了 *kconfig*(1M) 和 *mk_kernel*(1M) 命令所使用的内核配置信息。

系统描述文件包含了如下信息:

- 一行指定系统文件的版本。
- 一个要配置的封装和传统的内核模块列表。
- 一些可调整系统参数的预设值，以及其他系统级的配置信息。

以星号 (*) 开始的行是注释行。HP 不鼓励向系统描述文件添加注释，因为内核配置命令不会保留这些注释。有关详细信息，请参阅 kconfig(5)。

HP-UX 系统描述文件

系统文件可能包含一行说明文字，指定写入文件时所使用的系统文件语法版本。如果存在此行说明，则它必须是系统描述文件的第一个非注释行。这是系统描述文件中必须占据特定位置的唯一一行。其他所有的行都是与位置无关的。此行的格式如下:

version *version*

其中， *version* 可以有如下取值:

1 系统文件按照此手册页所描述的方式进行格式化。

0 系统文件会按照 HP-UX 11i 1.6 及其以前版本所使用的方式进行格式化。如果系统文件中没有 **version** 行，则这个值就是缺省值。

新的系统文件应当总是使用第 1 版的系统文件语法进行创建。在以后的 HP-UX 版本中将删除对第 0 版的支持。

由内核配置命令所产生的系统文件会包含一行指示所描述配置的信息。此行的形式如下:

configuration *name title timestamp*

where:

name 是配置的名字，

title 是配置的标题（在引号中），并且

timestamp 指出系统文件的创建时间。

系统文件可以包含一些行来列出要配置的传统的和封装好的内核模块（包含设备驱动程序和伪设备驱动程序）。

这样的行具有下面两种格式之一:

module 其中， *module* 可以是一个传统的或者是一个封装好的内核模块名称。

例如， **scsi** 选择 SCSI 硬盘驱动器的驱动程序， **scsitape** 选择 SCSI 磁带机的驱动程序，而 **nfs** 选择 NFS 子系统。此行的格式是为了实现向后兼容。它不允许指定模块的状态；模块的状态是模块开发者提供的缺省状态。要指定模块的期望状态，请使用第二种格式。

module *module* *state* [*version*]

其中， *module* 可以是一个传统的或者封装好的模块名称。 *state* 是 **best** 之一， **static** 、 **auto** 或者 **loaded** ；请参阅 *kcmodule*(1M) 以获取这些状态的定义。

version 是模块的版本。版本字段是可选的。当手动编辑或者创建系统文件时，通常要忽略该字段。当克隆内核配置时， **kconfig -e** 和 **kconfig -i** 会使用此字段；有关详细信息，请参阅 *kconfig*(1M) 。

例如，

module scsi static [3E0D0C96]

选择 SCSI 磁盘驱动器的驱动程序。

module pseudodrv loaded 0.1.0

动态加载 pseudodrv 模块。

系统文件还可能包含用于以下方面的行：

- 定义交换设备
- 定义转储设备
- 将一个驱动程序显式地绑定到硬件路径上
- 定义所选系统参数的状态和值。

对每个类别，按照如下所示创建每一行。

- 交换设备规范

至多只能有一个交换规范。如果没有给出交换规范，则主交换绑定会设置为 **Default** 。在使用逻辑卷管理器 (LVM) 的系统中， **Default** 等价于 **lvol** 。在其他系统中， **Default** 要标记于根磁盘，在根文件系统的末尾和磁盘末尾之间的区域。

swap *hw_path* *offset* [*blocks*]

按照指定的值配置交换设备位置和其大小。参数解释如下：

hw_path 表示将作为交换设备或者字符串缺省值而配置的设备的硬件路径，可能用于指出正在使用根设备。

offset 交换区位置。界限位于 1K 字节间隔上。负值表明在设备上希望有一个文件系统。在启动时，系统读取超级块以获得文件系统的准确大小，而且这个值会放在 *offset* 中。当交换分区是自动配置时，会采用这个机制。如果一个超级块是无效的，该条目就会被跳过，这样一个损坏的超级块不会由于要在其上配置交换分区而在后面导致整个文件系统的损坏。 *offset* 取正值或者零，表示需要保留的最小区域。零值意味着在设备的开始部分不保留任何值。零值意味着设备上没有文件

系统。

blocks 交换分区上大小为 1K 字节的磁盘块的个数（按十进制计数）。对于此交换设备规范，只有 *blocks* 参数是可选的。零是自动配置的缺省值。如果 *blocks* 为零，则设备剩下的所有部分会作为交换区域自动配置。如果 *blocks* 取非零值，则它的绝对值作为交换区域大小的上界。然后，如果交换区域实际上减少了，则 *blocks* 的符号决定了 *blocks* 是否保持不变，这时交换区域与保留区相邻，或者 *blocks* 被未使用区域的大小充满，从而使交换区域与设备的尾部相邻。

swap lvol 在逻辑卷上配置交换。

swap none 不使用交换设备配置内核。

- 转储设备规范

可以有一个或者多个转储规范。如果没有给出转储规范，则使用主交换区域。

dump *hw_path* 按照指定的设置配置转储设备的位置和大小。 *hw_path* 为设备的硬件路径，此路径会作为转储设备配置，否则字符串缺省值可能会用于指定使用主交换区域。

dump lvol 在逻辑卷上配置转储。

dump none 不使用转储设备配置内核。

- 显式设备驱动程序绑定

硬件路径规范可以有一个或者多个驱动程序。如果提供了一个驱动程序语句，那么指定的硬件模块会被强制加入给定硬件路径上的内核 I/O 系统。这可用于使系统识别一个无法被自动识别的设备。

driver *hw_path driver_name*

将驱动程序绑定到给定的硬件路径上的内核 I/O 系统。参数解释如下:

hw_path 表示软件要绑定的设备硬件路径。

driver_name 要在指定的硬件路径上绑定到内核的软件模块名称。

- 内核可调整参数设置

这些行包含了用于内核配置的可调整参数值（非缺省值）。一个可调整值可以包含一个数字或者公式，但是它不能包含空格。如果可调整参数是用户定义的，那么在 *parameter_name* 前面会有一个关键字 **user:**. 不允许 **user:** 和 *parameter_name* 之间有空格出现。每行都有如下形式:

[**tunable**] [**user:**]*parameter_name value*

模块化系统文件

模块化系统文件已从 HP-UX 配置范例中删除。所有模块化系统文件所需的信息都已经被合并到传统的系统文件中，于是创建了一个单一的 hpux 系统描述文件。

文件

/stand/system 用于 **nextboot** 内核配置的 HP-UX 系统描述文件

/stand/*config***/system** 名为 *config* 用于内核配置的 HP-UNIX 系统描述文件

另请参阅

kconfig(1M)、 kctune(1M)、 mk_kernel(1M) 和 kconfig(5)。

名称

tar - tar 磁带归档的格式

说明

由 **tar** （请参阅 *tar*(1)）产生的 *header* 结构如下（右边显示了常量定义的数组大小）：

```
struct {
    char name[NAMSIZ];              (100)
    char mode[MODE_SZ];             (8)
    char uid[UID_SZ];               (8)
    char gid[GID_SZ];               (8)
    char size[SIZE_SZ];             (12)
    char mtime[MTIME_SZ];           (12)
    char chksum[CHKSUM_SZ];         (8)
    char typeflag;
    char linkname[NAMSIZ];          (100)
    char magic[MAGIC_SZ];           (6)
    char version[VERSION_SZ];       (2)
    char uname[UNAME_SZ];           (32)
    char gname[GNAME_SZ];           (32)
    char devmajor[DEV_SZ];          (8)
    char devminor[DEV_SZ];          (8)
    char prefix[PREFIX_SZ];         (155)
} dbuf;
```

所有字符都用 ASCII 表示。标题块中没有使用填充；所有字段都是连续的。

字段 *magic* 、 *uname* 和 *gname* 是以空字符结尾的字符串。除非数组中的所有字符（包括最后一个字符），包含非空字符，字段 *name* 、 *linkname* 和 *prefix* 是以空字符结尾的字符串。 *version* 字段是包含字符 **00** （零零）的两个字节。 *typeflag* 包含单个字符。所有其他字段都是前面补零的 ASCII 八进制数。每个数字字段以一个或多个空格或空字符结尾。

name 和 *prefix* 字段生成该文件的路径名。该文件层次关系的保留，通过将路径名指定为路径前缀并将一个斜线字符和文件名作为后缀。如果 *prefix* 包含非空字符，则将 *prefix* 与一个斜线字符和 *name* 串连起来而无须修改或添加新字符来生成新路径名。以这种方式，可以支持最多 256 个字符的路径名。如果路径名不适合提供的空间，则格式创建实用程序通知用户该错误，同时不会尝试在介质上保存文件、标题或数据的任何部分。

另请参阅

tar(1)

符合的标准

tar: XPG4、FIPS 151-2、POSIX.1

名称

tcpd.conf - tcpd 的配置文件

说明

当 **inetd** 为一个服务调用 **tcpd** 时，将读取 **/etc/tcpd.conf** 并执行访问控制检查（请参阅 *tcpd*(1M)）。

文件中的每一行可当作注释或配置信息进行处理。注释行以 **#** 开头。未注释的行包含两个必需的字段，键和值。各字段之间用制表符和（或）空格分隔。如果一行以反斜线 (\) 终止，则该行可继续。

下列是配置参数：

rfc931_timeout *n*

通过此参数可启用或禁用 RFC931 用户名查找。 *n* 的值指定超时值（单位为秒），它将在从客户端获取用户名时使用。

n 的值为零，将禁用 rfc931 功能。

其缺省配置使用等于 0 的 *n* 值来禁用 rfc931 功能。

n 可设置的最大值是 30 秒。

on_reverselookup_fail {**allow** | **deny**}

此参数决定在反向查找失败时 **tcpd** 应允许还是拒绝连接请求。

在这两种情况下， **tcpd** 将记录反向查找失败事件，但在 **deny** 情况下，它将在反向查找失败之后拒绝连接请求。在 **allow** 情况下，主机名可与访问文件（ /etc/hosts.allow 和 **/etc/hosts.deny** ）中的 PARANOID 通配符相匹配（请参阅 *hosts_options*(5)）。

其缺省值为 **deny** 。

log_level {**normal** | **extended**}

此参数决定 **tcpd** 使用 **syslog** 应记录信息的级别。 **extended** 值将使 TCP 包装守护程序（请参阅 *tcpd*(1M)）记录 ACL 信息，比如匹配客户端请求的哪个条目，以及此条目的相关选项。

此条目的缺省值是 **normal** ，在这种情况下， **tcpd** 将以“connection from abc@xyz_host”的形式记录关于连接拒绝或接受的连接详细信息。

处理无效的和多个条目

tcpd 以下列方式处理无效的条目以及多个条目：

- 忽略配置参数的无效条目。将使用配置参数的缺省值作为替代。例如， *log_level* 的下述无效条目将通过使用 *normal* 来替换。

  log_level abcd

  将被视为：

log_level normal

- 如果为一个配置参数指定多个条目，则仅处理最后出现的条目，并忽略余下的条目。例如，在 **rfc931_timeout** 的下列两个条目中，最后的值 25 是供该参数使用的。

  rfc931_timeout 10

  rfc931_timeout 25

举例

要为 RFC931 用户名查找设置一个 25 秒的超时值:

rfc931_timeout 25

要禁用 RFC931 用户名查找，请执行:

rfc931_timeout 0

要使 **tcpd** 允许一个主机反向查找失败，并在 ACL 中将主机作为 **PARANOID** 来处理:

on_reverselookup_fail allow

要设置扩展的记录选项，请执行:

log_level extended

作者

tcpd.conf 由 HP 开发。

另请参阅

inetd(1M)、 tcpd(1M)。

名称

term - 编译的术语文件格式

概要

term

说明

编译的 terminfo 描述放置在目录 **/usr/share/lib/terminfo** 下。要想避免线性搜索很大的 HP-UX 系统目录，可使用二级方案： **/usr/share/lib/terminfo/***c***/***name* ，其中 *name* 是终端的名称，而 *c* 是 *name* 的第一个字符。因此，可在文件 **/usr/share/lib/terminfo/h/hp110** 中找到 **hp110** 。对相同终端的同义名称由指向同一编译文件的多重链接来实现。

选择格式，使其与所有硬件上的格式相同。假定这是一个 8 位或更长的字节，但不假定字节排序或符号扩展。

编译文件由 **tic** 程序（参阅 *tic*(1M) ）创建，并由 **setupterm()** 例行程序读取。文件分为下述六个部分：

1. 文件的开头是标题部分，它包含具有下述格式的六个短整数：

   1. 幻数（八进制的 0432）；
   2. 名称部分的大小，以字节为单位；
   3. 布尔型部分的字节数；
   4. 数字部分中短整数的数量；
   5. 字符串部分中偏移量的数量（短整数）；
   6. 字符串表的大小，以字节为单位。

   短整数存储在 8 位字节中。第一个字节中包含值中最不重要的 8 个位；第二个字节中包含最重要的 8 个位。（因此，这里表示的值为 256∗ 第二个字节 + 第一个字节）。值 −1 由 **0377** ， **0377** 表示；其他负值都是非法的。 −1 通常意味着此终端中缺少一种功能。注意，此格式对应于 VAX 和 PDP-11 的硬件。而此格式与其硬件不对应的计算机，将这些整数作为两个字节读取，然后计算结果。

2. 接下来是终端名称部分。它包括 terminfo 描述的第一行，列出该终端的各个名称，并使用 **|** 字符来分隔这些名称。此部分以一个 ASCII NUL 字符结束。

3. 在布尔型部分中，每一个布尔型标志都有一个字节。此字节可以是 **0** 或 **1** ，分别表示该标志不存在或存在。这些功能的顺序，与它们在文件 **<term.h>** 中列出的顺序相同。

   如有必要，将在布尔型部分和数字部分之间插入一个空字节，从而确保数字部分从一个偶数字节开始。所有的短整数都在短单词边界上对齐。

4. 数字部分与标志部分类似。每个功能都包含两个字节，并存储为一个短整数。如果值为 −1，则认为此功能不存在。

5. 字符串部分也类似。每个功能都按照上面的格式存储为一个短整数。值为 −1 则意味着该功能不存在。否则，该值将被视作从字符串表开头的一个偏移量。 **^***X* 或 *****c* 格式中的特殊字符以它们的解释形式而非打印形式存储。填充信息 **$***nn* 和参数信息 **%***x* 按未解释的形式原样存储。

6. 最后一部分是字符串表。它包含在字符串部分中引用的所有字符串功能的值。每一个字符串都以空结尾。

注意， **setupterm()** 的功能集有可能会与当前文件中的功能集不同。自 **setupterm()** 重新编译以来，数据库有可能已经更新（导致文件中额外的无法识别的条目），或是程序在最近数据库更新以后已经重新编译（导致缺失多个条目）。例行程序 **setupterm()** 必须为这两种可能性做好准备，这也就是为什么要包含数量和大小的原因。而且，新功能总是必须要添加在布尔型、数字和字符串功能列表的结尾。

下面这个例子是 HP 便携式计算机 (HP-110) 的一个八进制的描述转储：

```
110|hp110|hp110a portable computer,
    am, xhp, da, db, mir, cols#80, lines#16, lm#0,
    cbt=\Ei, bel=^G, cr=\r, tbc=\E3, clear=\E&a0y0C\EJ,
    el=\EK, ed=\EJ, hpa=\E&a%p1%dC, cup=\E&a%p1%dy%p2%dC,
    cud1=\EB, cub1=\b, cuf1=\EC, cuu1=\EA, cvvis=\E&j@,
    dch1=\EP, dl1=\EM, smir=\EQ, smso=\E&dB, sgr0=\E&d@,
    rmir=\ER, rmso=\E&d@, is2=\E&j@,
    if=/usr/share/lib/tabset/stdcrt, il1=\EL, kbs=\b, kcud1=\EB,
    khome=\Eh, kcub1=\ED, kcuf1=\EC, kcuu1=\EA, rmkx=\E&s0A,
    smkx=\E&s1A, vpa=\E&a%p1%dY, ind=\n, hts=\E1, ht=\t,
```

```
0000 032 001   #  \0 025  \0  \b  \0 223  \0 254  \0   1   1   0   |
0020   h   p   1   1   0   |   h   p   1   1   0   a       p   o   r
0040   t   a   b   l   e       c   o   m   p   u   t   e   r  \0  \0
0060 001  \0 001  \0  \0  \0  \0  \0  \0  \0 001 001 001  \0  \0  \0
0100  \0  \0  \0  \0   P  \0 377 377 020  \0  \0  \0 377 377 377 377
0120 377 377 377 377  \0  \0 003  \0 005  \0 377 377 007  \0  \n  \0
0140 024  \0 027  \0 032  \0 377 377   $  \0   4  \0 377 377 377 377
0160   7  \0 377 377 377 377   9  \0 377 377   <  \0   ?  \0   D  \0
0200   G  \0 377 377 377 377 377 377 377 377 377 377 377 377 377 377
0220 377 377   J  \0 377 377 377 377 377 377   M  \0 377 377 377 377
0240 377 377   R  \0 377 377 377 377   W  \0   Z  \0 377 377 377 377
0260 377 377 377 377 377 377   _  \0 377 377   d  \0 377 377   {  \0
0300 377 377   ~  \0 377 377 377 377 377 377 377 377 377 377 200  \0
0320 377 377 377 377 377 377 377 377 377 377 377 377 377 377 377 377
0340 377 377 377 377 377 377 377 377 377 377 377 377 203  \0 377 377
0360 377 377 206  \0 377 377 377 377 377 377 211  \0 377 377 377 377
0400 377 377 214  \0 217  \0 225  \0 377 377 377 377 377 377 377 377
0420 377 377 377 377 377 377 377 377 377 377 377 377 377 377 377 377

0520 377 377 233  \0 377 377 245  \0 377 377 377 377 247  \0 377 377
```

```
0540 252  \0 377 377 377 377 377 377 377 377 377 377 377 377 377 377
0560 377 377 377 377 377 377 377 377 377 377 033   i  \0 007  \0  \r
0600  \0 033   3  \0 033   &   a   0   y   0   C 033   J  \0 033   K
0620  \0 033   J  \0 033   &   a   %   p   1   %   d   C  \0 033   &
0640   a   %   p   1   %   d   y   %   p   2   %   d   C  \0 033   B
0660  \0  \b  \0 033   C  \0 033   A  \0 033   &   j   @  \0 033   P
0700  \0 033   M  \0 033   Q  \0 033   &   d   B  \0 033   &   d   @
0720  \0 033   R  \0 033   &   d   @  \0 033   &   j   @  \0   /   u
0740   s   r   /   l   i   b   /   t   a   b   s   e   t   /   s   t
0760   d   c   r   t  \0 033   L  \0  \b  \0 033   B  \0 033   h  \0
1000 033   D  \0 033   C  \0 033   A  \0 033   &   s   0   A  \0 033
1020   &   s   1   A  \0 033   &   a   %   p   1   %   d   Y  \0  \n
1040  \0 033   1  \0  \t  \0
1046
```

警告

编译条目的大小总计不能超过 4096 字节。

名称字段不能超过 128 字节。

HP Company 只支持在当前受支持的设备列表中列出的终端。然而，不支持的和支持的终端都可位于 terminfo 数据库中。如果使用不支持的终端，则它们不能够正常工作。

文件

/usr/share/lib/terminfo/?/* 编译的终端功能数据库

另请参阅

tic(1M)、 untic(1M)、 terminfo(4)。

名称

term_c：term.h、TERM - 终端功能

说明

头 **<term.h>** 包含下列各表中每个符号常量和功能名称的定义。

在下表中， 变量是 **C** 程序员用来（以 **terminfo** 级别）访问某个功能的名称。 功能名是在 **terminfo** 源文件中指定的功能的简称。更新源文件的人可以使用它，通过 **tput** 命令也可以使用它。

布尔值

变量	功能 名称	终端功能 代码	说明
auto_left_margin	***bw***	***bw***	**cub1** 从第 0 列自动分行到最后一列
auto_right_margin	***am***	***am***	终端有自动边距
back_color_erase	***bce***	***ut***	屏幕以背景色清除
buttons	***btns***	***BT***	鼠标上按钮的数量
can_change	***ccc***	***cc***	终端可以重新定义现有的颜色
ceol_standout_glitch	***xhp***	***xs***	突出内容不会被覆盖功能清除 (hp)
col_addr_glitch	***xhpa***	***YA***	对于 **hpa/mhpa** 功能而言只做正向移动
cpi_changes_res	***cpix***	***YF***	更改字符间距会改变分辨率
create_window	***cwin***	***CW***	将窗口 #1 定义为从 #2、#3 到 #4、#5
cr_cancels_micro_mode	***crxm***	***YB***	使用 **cr** 关闭微模式
dest_tabs_magic_smso	***xt***	***xt***	破坏性制表符，magic **smso** 字符 (t1061)
dial_phone	***dial***	***DI***	拨打电话号码 #1
display_clock	***dclk***	***DK***	显示时钟
eat_newline_glitch	***xenl***	***xn***	在 80 列之后忽略换行符 (Concept)
erase_overstrike	***eo***	***eo***	可以用空格来清除叠印
fixed_pause	***pause***	***PA***	暂停 2-3 秒
flash_hook	***hook***	***fh***	拍叉簧
generic_type	***gn***	***gn***	常规线路类型（例如，拨号线路、交换线路）
get_mouse	***getm***	***Gm***	Curses 应当获得按钮事件
goto_window	***wingo***	***WG***	转至窗口 #1
hangup	***hup***	***HU***	挂断电话
hard_copy	***hc***	***hc***	硬拷贝终端
hard_cursor	***chts***	***HC***	难以看到光标
has_meta_key	***km***	***km***	有一个元键（Shift，设置奇偶校验位）
has_print_wheel	***daisy***	***YC***	打印机需要操作员来更改字符集
has_status_line	***hs***	***hs***	有一个额外的“状态行”
hue_lightness_saturation	***hls***	***hl***	终端只使用 HLS 颜色表示法 (Tektronix)

insert_null_glitch	***in***	***in***	插入模式，能识别空行
lpi_changes_res	***lpix***	***YG***	更改行间距会改变分辨率
memory_above	***da***	***da***	显示可以保留在屏幕上方
memory_below	***db***	***db***	显示可以保留在屏幕下方
move_insert_mode	***mir***	***mi***	在插入模式下可以安全地移动
move_standout_mode	***msgr***	***ms***	在突出模式下可以安全地移动
needs_xon_xoff	***nxon***	***nx***	不能填充，需要 xon/xoff
no_esc_ctlc	***xsb***	***xb***	Beehive（F1=Escape，F2=Ctrl C）
no_pad_char	***npc***	***NP***	填充字符不存在
non_dest_scroll_region	***ndscr***	***ND***	滚动区域不具有破坏性
non_rev_rmcup	***nrrmc***	***NR***	**smcup** 不会反转 **rmcup**
over_strike	***os***	***os***	终端可以在硬拷贝终端上叠印
print_rate	***cps***	***Ym***	打印速度（以每秒字符数为单位）
prtr_silent	***mc5i***	***5i***	打印机将不在屏幕上回显
row_addr_glitch	***xvpa***	***YD***	对于 **vpa/mvpa** 功能而言只做正向移动
semi_auto_right_margin	***sam***	***YE***	打印在最后一列将导致 **cr**
set_pglen_inch	***slength***	***YI***	将页长设置为一英寸的百分之 #1（使用 tparm）
status_line_esc_ok	***eslok***	***es***	可以在状态行上使用转义符
tilde_glitch	***hz***	***hz***	Hazeltine；不能打印波浪符 (˜)
transparent_underline	***ul***	***ul***	下划线字符叠印
xon_xoff	***xon***	***xo***	终端使用 xon/xoff 握手机制

数字

变量	功能 名称	终端功能 代码	说明
bit_image_entwining	***bitwin***	***Yo***	每个位映射行的运行次数
bit_image_type	***bitype***	***Yp***	位图设备的类型
buffer_capacity	***bufsz***	***Ya***	在打印之前缓冲的字节数
columns	***cols***	***co***	一行中的列数
dot_horz_spacing	***spinh***	***Yc***	在水平方向上点与点之间的距离（以每英寸点数为单位）
dot_vert_spacing	***spinv***	***Yb***	在垂直方向上针与针之间的距离（以每英寸针数为单位）
init_tabs	***it***	***it***	最初每隔 **#** 个空格有多少制表符
label_height	***lh***	***lh***	每个标签中的行数
label_width	***lw***	***lw***	每个标签中的列数
lines	***lines***	***li***	屏幕或页上的行数
lines_of_memory	***lm***	***lm***	大于 **lines** 时内存中的行数；**0** 表示可变
max_attributes	***ma***	***ma***	终端最多可以显示的组合视频属性

magic_cookie_glitch	***xmc***	***sg***	由 **smso** 或 **rmso** 留下的空白字符的数量
max_colors	***colors***	***Co***	屏幕上的最大颜色数
max_micro_address	***maddr***	***Yd***	**micro_..._address** 中的最大值
max_micro_jump	***mjump***	***Ye***	**parm_..._micro** 中的最大值
max_pairs	***pairs***	***pa***	屏幕上颜色对的最大数量
maximum_windows	***Wnum***	***MW***	可定义的窗口的最大数量
micro_char_size	***mcs***	***Yg***	在微模式下字符步长
micro_line_size	***mls***	***Yf***	在微模式下行步长
no_color_video	***ncv***	***NC***	不能与彩色配合使用的视频属性
num_labels	***nlab***	***Nl***	屏幕上标签的数量（从 1 开始）
number_of_pins	***npins***	***Yh***	打印头中的针数
output_res_char	***orc***	***Yi***	水平分辨率（以每字符单元数为单位）
output_res_line	***orl***	***Yj***	垂直分辨率（以每行单元数为单位）
output_res_horz_inch	***orhi***	***Yk***	水平分辨率（以每英寸单元数为单位）
output_res_vert_inch	***orvi***	***Yl***	垂直分辨率（以每英寸单元数为单位）
padding_baud_rate	***pb***	***pb***	需要填充时的最低波特率
virtual_terminal	***vt***	***vt***	虚拟终端号
wide_char_size	***widcs***	***Yn***	在双宽模式下字符步长
width_status_line	***wsl***	***ws***	状态行中的列数

字符串

变量	功能 名称	终端功能 代码	说明
acs_chars	***acsc***	***ac***	图形字符集对 aAbBcC
alt_scancode_esc	***scesa***	***S8***	用于扫描代码模拟的备用转义符（缺省值适用于 vt100）
back_tab	***cbt***	***bt***	向后 tab
bell	***bel***	***bl***	声音信号（铃声）
bit_image_carriage_return	***bicr***	***Yv***	移到同一行的开头（使用 tparm）
bit_image_newline	***binel***	***Zz***	移到位图的下一行（使用 tparm）
bit_image_repeat	***birep***	***Xy***	将位图单元 #1 重复 #2 次（使用 tparm）
carriage_return	***cr***	***cr***	回车
change_char_pitch	***cpi***	***ZA***	更改每英寸字符数
change_line_pitch	***lpi***	***ZB***	更改每英寸行数
change_res_horz	***chr***	***ZC***	更改水平分辨率
change_res_vert	***cvr***	***ZD***	更改垂直分辨率
change_scroll_region	***csr***	***cs***	更改为第 #1 行至第 #2 行 (vt100)
char_padding	***rmp***	***rP***	与 **ip** 类似，但在替换模式下使用

char_set_names	***csnm***	***Zy***	字符集名称的列表
clear_all_tabs	***tbc***	***ct***	清除所有制表位
clear_margins	***mgc***	***MC***	清除所有边距（上边距、下边距、左边距和右边距）
clear_screen	***clear***	***cl***	清除屏幕并使光标回到起始点
clr_bol	***el1***	***cb***	清除到行首（包括行首）
clr_eol	***el***	***ce***	清除到行尾
clr_eos	***ed***	***cd***	清除到显示内容的末尾
code_set_init	***csin***	***ci***	多个代码集的初始化序列
color_names	***colornm***	***Yw***	为颜色 #1 提供名称
column_address	***hpa***	***ch***	水平绝对位置
command_character	***cmdch***	***CC***	原型中可由终端设置的 cmd 字符
cursor_address	***cup***	***cm***	移到第 #1 行第 #2 列
cursor_down	***cud1***	***do***	下移一行
cursor_home	***home***	***ho***	使光标回到起始点（如果没有 **cup**）
cursor_invisible	***civis***	***vi***	使光标不可见
cursor_left	***cub1***	***le***	向左移动一个空格。
cursor_mem_address	***mrcup***	***CM***	内存的相对光标寻址
cursor_normal	***cnorm***	***ve***	使光标以正常方式显示（撤消 **vs/vi**）
cursor_right	***cuf1***	***nd***	非破坏性空格（光标或回车右移）
cursor_to_ll	***ll***	***ll***	最后一行，第一列（如果没有 **cup**）
cursor_up	***cuu1***	***up***	Upline（光标上移）
cursor_visible	***cvvis***	***vs***	使光标可见
define_bit_image_region	***defbi***	***Yx***	定义矩形位图区域（使用 tparm）
define_char	***defc***	***ZE***	定义字符集中的字符
delete_character	***dch1***	***dc***	删除字符
delete_line	***dl1***	***dl***	删除行
device_type	***devt***	***dv***	指明语言（或代码集）支持
dis_status_line	***dsl***	***ds***	禁用状态行
display_pc_char	***dispc***	***S1***	显示 PC 字符
down_half_line	***hd***	***hd***	下移半行（向前换 1/2 行）
ena_acs	***enacs***	***eA***	启用备用字符集
end_bit_image_region	***endbi***	***Yy***	结束位图区域（使用 tparm）
enter_alt_charset_mode	***smacs***	***as***	启动备用字符集
enter_am_mode	***smam***	***SA***	打开自动边距功能
enter_blink_mode	***blink***	***mb***	打开闪烁功能
enter_bold_mode	***bold***	***md***	打开粗体（超亮）模式
enter_ca_mode	***smcup***	***ti***	启动使用 **cup** 的程序的字符串
enter_delete_mode	***smdc***	***dm***	删除模式（进入）

enter_dim_mode	*dim*	*mh*	打开半亮模式
enter_doublewide_mode	*swidm*	*ZF*	启用双宽打印
enter_draft_quality	*sdrfq*	*ZG*	设置草稿质量打印
enter_horizontal_hl_mode	*ehhlm*	*N/A*	打开水平突出显示模式
enter_insert_mode	*smir*	*im*	插入模式（进入）
enter_italics_mode	*sitm*	*ZH*	启用斜体
enter_left_hl_mode	*elhlm*	*N/A*	打开左突出显示模式
enter_leftward_mode	*slm*	*ZI*	启用向左移动回车功能
enter_low_hl_mode	*elohlm*	*N/A*	打开低突出显示模式
enter_micro_mode	*smicm*	*ZJ*	启用微移动功能
enter_near_letter_quality	*snlq*	*ZK*	设置仿信函质量打印
enter_normal_quality	*snrmq*	*ZL*	设置普通质量打印
enter_pc_charset_mode	*smpch*	*S2*	进入 PC 字符显示模式
enter_protected_mode	*prot*	*mp*	打开受保护模式
enter_reverse_mode	*rev*	*mr*	打开反转视频模式
enter_right_hl_mode	*erhlm*	*N/A*	打开右突出显示模式
enter_scancode_mode	*smsc*	*S4*	进入 PC 扫描代码模式
enter_secure_mode	*invis*	*mk*	打开空白模式（字符不可见）
enter_shadow_mode	*sshm*	*ZM*	启用阴影打印
enter_standout_mode	*smso*	*so*	启动突出模式
enter_subscript_mode	*ssubm*	*ZN*	启用下标打印
enter_superscript_mode	*ssupm*	*ZO*	启用上标打印
enter_top_hl_mode	*ethlm*	*N/A*	打开上突出显示模式
enter_underline_mode	*smul*	*us*	启动下划线模式
enter_upward_mode	*sum*	*ZP*	启用向上移动回车功能
enter_vertical_hl_mode	*evhlm*	*N/A*	打开垂直突出显示模式
enter_xon_mode	*smxon*	*SX*	打开 xon/xoff 握手机制
erase_chars	*ech*	*ec*	清除 #1 个字符
exit_alt_charset_mode	*rmacs*	*ae*	结束备用字符集
exit_am_mode	*rmam*	*RA*	关闭自动边距功能
exit_attribute_mode	*sgr0*	*me*	关闭所有属性
exit_ca_mode	*rmcup*	*te*	终止使用 **cup** 的程序的字符串
exit_delete_mode	*rmdc*	*ed*	结束删除模式
exit_doublewide_mode	*rwidm*	*ZQ*	禁用双宽打印
exit_insert_mode	*rmir*	*ei*	结束插入模式
exit_italics_mode	*ritm*	*ZR*	禁用斜体
exit_leftward_mode	*rlm*	*ZS*	启用向右（普通）移动回车功能
exit_micro_mode	*rmicm*	*ZT*	禁用微移动功能

exit_pc_charset_mode	***rmpch***	***S3***	禁用 PC 字符显示模式
exit_scancode_mode	***rmsc***	***S5***	禁用 PC 扫描代码模式
exit_shadow_mode	***rshm***	***ZU***	禁用阴影打印
exit_standout_mode	***rmso***	***se***	结束突出模式
exit_subscript_mode	***rsubm***	***ZV***	禁用下标打印
exit_superscript_mode	***rsupm***	***ZW***	禁用上标打印
exit_underline_mode	***rmul***	***ue***	结束下划线模式
exit_upward_mode	***rum***	***ZX***	启用向下（普通）移动回车功能
exit_xon_mode	***rmxon***	***RX***	关闭 xon/xoff 握手机制
flash_screen	***flash***	***vb***	可视铃声（不能移动光标）
form_feed	***ff***	***ff***	弹出硬拷贝终端页
from_status_line	***fsl***	***fs***	从状态行返回
init_1string	***is1***	***i1***	终端或打印机初始化字符串
init_2string	***is2***	***is***	终端或打印机初始化字符串
init_3string	***is3***	***i3***	终端或打印机初始化字符串
init_file	***if***	***if***	初始化文件的名称
init_prog	***iprog***	***iP***	要初始化的程序的路径名
initialize_color	***initc***	***IC***	对颜色定义进行初始化
initialize_pair	***initp***	***Ip***	对颜色对进行初始化
insert_character	***ich1***	***ic***	插入字符
insert_line	***il1***	***al***	添加新空白行
insert_padding	***ip***	***ip***	在插入的字符之后插入填充字符

key_ 字符串由特定键发送。 **key_** 说明包括在按下该键时由 CURSES 函数 **getch()** （请参阅 *getch*(3X) ）返回的代码的宏，该宏在 **<curses.h>** 中定义。

变量	功能 名称	终端功能 代码	说明
key_a1	***ka1***	***K1***	小键盘左上角的键
key_a3	***ka3***	***K3***	小键盘右上角的键
key_b2	***kb2***	***K2***	小键盘中央的键
key_backspace	***kbs***	***kb***	由退格键发送
key_beg	***kbeg***	***@1***	由开始键发送
key_btab	***kcbt***	***kB***	由向后 tab 键发送
key_c1	***kc1***	***K4***	小键盘左下角的键
key_c3	***kc3***	***K5***	小键盘右下角的键
key_cancel	***kcan***	***@2***	由取消键发送

key_catab	*ktbc*	*ka*	由清除所有制表符键发送
key_clear	*kclr*	*kC*	由清除屏幕或清除键发送
key_close	*kclo*	*@3*	由关闭键发送
key_command	*kcmd*	*@4*	由 cmd（命令）键发送
key_copy	*kcpy*	*@5*	由复制键发送
key_create	*kcrt*	*@6*	由创建键发送
key_ctab	*kctab*	*kt*	由清除制表符键发送
key_dc	*kdch1*	*kD*	由删除字符键发送
key_dl	*kdl1*	*kL*	由删除行键发送
key_down	*kcud1*	*kd*	由终端向下键发送
key_eic	*krmir*	*kM*	在插入模式下由 **rmir** 或 **smir** 发送
key_end	*kend*	*@7*	由 End 键发送
key_enter	*kent*	*@8*	由 Enter（或发送）键发送
key_eol	*kel*	*kE*	由清除到行尾键发送
key_eos	*ked*	*kS*	由清除到屏幕末尾键发送
key_exit	*kext*	*@9*	由退出键发送
key_f0	*kf0*	*k0*	由功能键 F0 发送
key_f1	*kf1*	*k1*	由功能键 F1 发送
key_f2	*kf2*	*k2*	由功能键 F2 发送
key_f3	*kf3*	*k3*	由功能键 F3 发送
key_f4	*kf4*	*k4*	由功能键 F4 发送
key_f5	*kf5*	*k5*	由功能键 F5 发送
key_f6	*kf6*	*k6*	由功能键 F6 发送
key_f7	*kf7*	*k7*	由功能键 F7 发送
key_f8	*kf8*	*k8*	由功能键 F8 发送
key_f9	*kf9*	*k9*	由功能键 F9 发送
key_f10	*kf10*	*k;*	由功能键 F10 发送
key_f11	*kf11*	*F1*	由功能键 F11 发送
key_f12	*kf12*	*F2*	由功能键 F12 发送
key_f13	*kf13*	*F3*	由功能键 F13 发送
key_f14	*kf14*	*F4*	由功能键 F14 发送
key_f15	*kf15*	*F5*	由功能键 F15 发送
key_f16	*kf16*	*F6*	由功能键 F16 发送
key_f17	*kf17*	*F7*	由功能键 F17 发送
key_f18	*kf18*	*F8*	由功能键 F18 发送
key_f19	*kf19*	*F9*	由功能键 F19 发送
key_f20	*kf20*	*FA*	由功能键 F20 发送
key_f21	*kf21*	*FB*	由功能键 F21 发送

key_f22	***kf22***	***FC***	由功能键 F22 发送
key_f23	***kf23***	***FD***	由功能键 F23 发送
key_f24	***kf24***	***FE***	由功能键 F24 发送
key_f25	***kf25***	***FF***	由功能键 F25 发送
key_f26	***kf26***	***FG***	由功能键 F26 发送
key_f27	***kf27***	***FH***	由功能键 F27 发送
key_f28	***kf28***	***FI***	由功能键 F28 发送
key_f29	***kf29***	***FJ***	由功能键 F29 发送
key_f30	***kf30***	***FK***	由功能键 F30 发送
key_f31	***kf31***	***FL***	由功能键 F31 发送
key_f32	***kf32***	***FM***	由功能键 F32 发送
key_f33	***kf33***	***FN***	由功能键 F33 发送
key_f34	***kf34***	***FO***	由功能键 F34 发送
key_f35	***kf35***	***FP***	由功能键 F35 发送
key_f36	***kf36***	***FQ***	由功能键 F36 发送
key_f37	***kf37***	***FR***	由功能键 F37 发送
key_f38	***kf38***	***FS***	由功能键 F38 发送
key_f39	***kf39***	***FT***	由功能键 F39 发送
key_f40	***kf40***	***FU***	由功能键 F40 发送
key_f41	***kf41***	***FV***	由功能键 F41 发送
key_f42	***kf42***	***FW***	由功能键 F42 发送
key_f43	***kf43***	***FX***	由功能键 F43 发送
key_f44	***kf44***	***FY***	由功能键 F44 发送
key_f45	***kf45***	***FZ***	由功能键 F45 发送
key_f46	***kf46***	***Fa***	由功能键 F46 发送
key_f47	***kf47***	***Fb***	由功能键 F47 发送
key_f48	***kf48***	***Fc***	由功能键 F48 发送
key_f49	***kf49***	***Fd***	由功能键 F49 发送
key_f50	***kf50***	***Fe***	由功能键 F50 发送
key_f51	***kf51***	***Ff***	由功能键 F51 发送
key_f52	***kf52***	***Fg***	由功能键 F52 发送
key_f53	***kf53***	***Fh***	由功能键 F53 发送
key_f54	***kf54***	***Fi***	由功能键 F54 发送
key_f55	***kf55***	***Fj***	由功能键 F55 发送
key_f56	***kf56***	***Fk***	由功能键 F56 发送
key_f57	***kf57***	***Fl***	由功能键 F57 发送
key_f58	***kf58***	***Fm***	由功能键 F58 发送
key_f59	***kf59***	***Fn***	由功能键 F59 发送

key_f60	*kf60*	*Fo*	由功能键 F60 发送
key_f61	*kf61*	*Fp*	由功能键 F61 发送
key_f62	*kf62*	*Fq*	由功能键 F62 发送
key_f63	*kf63*	*Fr*	由功能键 F63 发送
key_find	*kfnd*	*@0*	由查找键发送
key_help	*khlp*	*%1*	由帮助键发送
key_home	*khome*	*kh*	由 Home 键发送
key_ic	*kich1*	*kI*	由插入字符（或进入插入模式）键发送
key_il	*kil1*	*kA*	由插入行键发送
key_left	*kcub1*	*kl*	由终端向左键发送
key_ll	*kll*	*kH*	由 home-down 键发送
key_mark	*kmrk*	*%2*	由标记键发送
key_message	*kmsg*	*%3*	由消息键发送
key_mouse	*kmous*	*Km*	0631，发生鼠标操作事件
key_move	*kmov*	*%4*	由移动键发送
key_next	*knxt*	*%5*	由下一个对象键发送
key_npage	*knp*	*kN*	由下一页键发送
key_open	*kopn*	*%6*	由打开键发送
key_options	*kopt*	*%7*	由选项键发送
key_ppage	*kpp*	*kP*	由上一页键发送
key_previous	*kprv*	*%8*	由上一个对象键发送
key_print	*kprt*	*%9*	由打印或复制键发送
key_redo	*krdo*	*%0*	由恢复键发送
key_reference	*kref*	*&1*	由引用键发送
key_refresh	*krfr*	*&2*	由刷新键发送
key_replace	*krpl*	*&3*	由替换键发送
key_restart	*krst*	*&4*	由重新启动键发送
key_resume	*kres*	*&5*	由继续键发送
key_right	*kcuf1*	*kr*	由终端向右键发送
key_save	*ksav*	*&6*	由保存键发送
key_sbeg	*kBEG*	*&9*	通过在按 Shift 键的同时按开始键来发送
key_scancel	*kCAN*	*&0*	通过在按 Shift 键的同时按取消键来发送
key_scommand	*kCMD*	**1*	通过在按 Shift 键的同时按命令键来发送
key_scopy	*kCPY*	**2*	通过在按 Shift 键的同时按复制键来发送
key_screate	*kCRT*	**3*	通过在按 Shift 键的同时按创建键来发送
key_sdc	*kDC*	**4*	通过在按 Shift 键的同时按删除字符键来发送
key_sdl	*kDL*	**5*	通过在按 Shift 键的同时按删除行键来发送
key_select	*kslt*	**6*	由选择键发送

key_send	***kEND***	****7***	通过在按 Shift 键的同时按 End 键来发送
key_seol	***kEOL***	****8***	通过在按 Shift 键的同时按清除行键来发送
key_sexit	***kEXT***	****9***	通过在按 Shift 键的同时按退出键来发送
key_sf	***kind***	***kF***	由向前滚动（或向下滚动）键发送
key_sfind	***kFND***	****0***	通过在按 Shift 键的同时按查找键来发送
key_shelp	***kHLP***	***#1***	通过在按 Shift 键的同时按帮助键来发送
key_shome	***kHOM***	***#2***	通过在按 Shift 键的同时按 Home 键来发送
key_sic	***kIC***	***#3***	通过在按 Shift 键的同时按输入键来发送
key_sleft	***kLFT***	***#4***	通过在按 Shift 键的同时按向左键来发送
key_smessage	***kMSG***	***%a***	通过在按 Shift 键的同时按消息键来发送
key_smove	***kMOV***	***%b***	通过在按 Shift 键的同时按移动键来发送
key_snext	***kNXT***	***%c***	通过在按 Shift 键的同时按下一个键来发送
key_soptions	***kOPT***	***%d***	通过在按 Shift 键的同时按选项键来发送
key_sprevious	***kPRV***	***%e***	通过在按 Shift 键的同时按上一个键来发送
key_sprint	***kPRT***	***%f***	通过在按 Shift 键的同时按打印键来发送
key_sr	***kri***	***kR***	由向后滚动（或向上滚动）键发送
key_sredo	***kRDO***	***%g***	通过在按 Shift 键的同时按恢复键来发送
key_sreplace	***kRPL***	***%h***	通过在按 Shift 键的同时按替换键来发送
key_sright	***kRIT***	***%i***	通过在按 Shift 键的同时按向右键来发送
key_srsume	***kRES***	***%j***	通过在按 Shift 键的同时按继续键来发送
key_ssave	***kSAV***	***!1***	通过在按 Shift 键的同时按保存键来发送
key_ssuspend	***kSPD***	***!2***	通过在按 Shift 键的同时按挂起键来发送
key_stab	***khts***	***kT***	由设置制表符键发送
key_sundo	***kUND***	***!3***	通过在按 Shift 键的同时按撤消键来发送
key_suspend	***kspd***	***&7***	由挂起键发送
key_undo	***kund***	***&8***	由撤消键发送
key_up	***kcuu1***	***ku***	由终端向上键发送
keypad_local	***rmkx***	***ke***	不处于“小键盘传输”模式
keypad_xmit	***smkx***	***ks***	将终端置于“小键盘传输”模式
lab_f0	***lf0***	***l0***	如果不是功能键 F0，则为 F0 上的标签
lab_f1	***lf1***	***l1***	如果不是功能键 F1，则为 F1 上的标签
lab_f2	***lf2***	***l2***	如果不是功能键 F2，则为 F2 上的标签
lab_f3	***lf3***	***l3***	如果不是功能键 F3，则为 F3 上的标签
lab_f4	***lf4***	***l4***	如果不是功能键 F4，则为 F4 上的标签
lab_f5	***lf5***	***l5***	如果不是功能键 F5，则为 F5 上的标签
lab_f6	***lf6***	***l6***	如果不是功能键 F6，则为 F6 上的标签
lab_f7	***lf7***	***l7***	如果不是功能键 F7，则为 F7 上的标签
lab_f8	***lf8***	***l8***	如果不是功能键 F8，则为 F8 上的标签

lab_f9	*lf9*	*l9*	如果不是功能键 F9，则为 F9 上的标签
lab_f10	*lf10*	*la*	如果不是功能键 F10，则为 F10 上的标签
label_format	*fln*	*Lf*	标签格式
label_off	*rmln*	*LF*	关闭软标签
label_on	*smln*	*LO*	打开软标签
meta_off	*rmm*	*mo*	关闭“元模式”
meta_on	*smm*	*mm*	打开“元模式”（第 8 位）
micro_column_address	*mhpa*	*ZY*	与微调整的 **column_address** 相似
micro_down	*mcud1*	*ZZ*	与微调整的 **cursor_down** 相似
micro_left	*mcub1*	*Za*	与微调整的 **cursor_left** 相似
micro_right	*mcuf1*	*Zb*	与微调整的 **cursor_right** 相似
micro_row_address	*mvpa*	*Zc*	与微调整的 **row_address** 相似
micro_up	*mcuu1*	*Zd*	与微调整的 **cursor_up** 相似
mouse_info	*minfo*	*Mi*	鼠标状态信息
newline	*nel*	*nw*	换行符（其行为与后跟 **lf** 的 **cr** 相似）
order_of_pins	*porder*	*Ze*	将软件位数与打印头针数相匹配
orig_colors	*oc*	*oc*	将所有颜色（或颜色对）设置为原始颜色（或颜色对）
orig_pair	*op*	*op*	将缺省的颜色对设置为原始颜色对
pad_char	*pad*	*pc*	填充字符（非空）
parm_dch	*dch*	*DC*	删除 #1 个字符
parm_delete_line	*dl*	*DL*	删除 #1 行
parm_down_cursor	*cud*	*DO*	向下移动 #1 行。
parm_down_micro	*mcud*	*Zf*	与微调整的 **parm_down_cursor** 相似。
parm_ich	*ich*	*IC*	插入 #1 个空白字符
parm_index	*indn*	*SF*	向前滚动 #1 行。
parm_insert_line	*il*	*AL*	添加 #1 个新空白行
parm_left_cursor	*cub*	*LE*	将光标向左移动 #1 个空格
parm_left_micro	*mcub*	*Zg*	与微调整的 **parm_left_cursor** 相似。
parm_right_cursor	*cuf*	*RI*	向右移动 #1 个空格。
parm_right_micro	*mcuf*	*Zh*	与微调整的 **parm_right_cursor** 相似。
parm_rindex	*rin*	*SR*	向后滚动 #1 行。
parm_up_cursor	*cuu*	*UP*	将光标向上移动 #1 行。
parm_up_micro	*mcuu*	*Zi*	与微调整的 **parm_up_cursor** 相似。
pc_term_options	*pctrm*	*S6*	PC 终端选项
pkey_key	*pfkey*	*pk*	对功能键 #1 进行设置以键入字符串 #2
pkey_local	*pfloc*	*pl*	对功能键 #1 进行设置以执行字符串 #2
pkey_plab	*pfxl*	*xl*	对键 #1 进行设置以传送字符串 #2 并显示字符串 #3
pkey_xmit	*pfx*	*px*	对功能键 #1 进行设置以传送字符串 #2

plab_norm	***pln***	***pn***	对标签 #1 进行设置以显示字符串 #2
print_screen	***mc0***	***ps***	打印屏幕内容
prtr_non	***mc5p***	***pO***	打开打印机，打印 #1 个字节
prtr_off	***mc4***	***pf***	关闭打印机
prtr_on	***mc5***	***po***	打开打印机
pulse	***pulse***	***PU***	选择脉冲拨号
quick_dial	***qdial***	***QD***	拨打电话号码 #1，不进行进度检测
remove_clock	***rmclk***	***RC***	删除时钟
repeat_char	***rep***	***rp***	将字符 #1 重复 #2 次
req_for_input	***rfi***	***RF***	发送下一个输入字符（对于 pty）
req_mouse_pos	***reqmp***	***RQ***	请求鼠标位置报告
reset_1string	***rs1***	***r1***	将终端完全重置为 sane 模式
reset_2string	***rs2***	***r2***	将终端完全重置为 sane 模式
reset_3string	***rs3***	***r3***	将终端完全重置为 sane 模式
reset_file	***rf***	***rf***	包含重置字符串的文件的名称
restore_cursor	***rc***	***rc***	将光标恢复到上一个屏幕的位置
row_address	***vpa***	***cv***	垂直绝对位置
save_cursor	***sc***	***sc***	保存光标位置
scancode_escape	***scesc***	***S7***	用于扫描代码模拟的转义符
scroll_forward	***ind***	***sf***	向上滚动文本
scroll_reverse	***ri***	***sr***	向下滚动文本
select_char_set	***scs***	***Zj***	选择字符集
set0_des_seq	***s0ds***	***s0***	切换到到代码集 0（EUC 集 0，ASCII）
set1_des_seq	***s1ds***	***s1***	切换到代码集 1
set2_des_seq	***s2ds***	***s2***	切换到代码集 2
set3_des_seq	***s3ds***	***s3***	切换到代码集 3
set_a_background	***setab***	***AB***	使用 ANSI 转义符设置背景色
set_a_foreground	***setaf***	***AF***	使用 ANSI 转义符设置前景色
set_attributes	***sgr***	***sa***	定义视频属性 #1-#9
set_background	***setb***	***Sb***	设置当前背景色
set_bottom_margin	***smgb***	***Zk***	设置当前行的下边距
set_bottom_margin_parm	***smgbp***	***Zl***	将下边距设置为距离底部 #1 或 #2 行
set_clock	***sclk***	***SC***	设置时钟
set_color_band	***setcolor***	***Yz***	更改为第 #1 号色带颜色
set_color_pair	***scp***	***sp***	设置当前的颜色对
set_foreground	***setf***	***Sf***	设置当前的前景色1
set_left_margin	***smgl***	***ML***	设置当前行的左边距
set_left_margin_parm	***smglp***	***Zm***	设置第 #1 (#2) 列的左（右）边距

set_lr_margin	***smglr***	***ML***	设置左边距和右边距
set_page_length	***slines***	***YZ***	将页长设置为 #1 行（使用 tparm）
set_right_margin	***smgr***	***MR***	设置当前列的右边距
set_right_margin_parm	***smgrp***	***Zn***	设置第 #1 列的右边距
set_tab	***hts***	***st***	在当前列的所有行设置制表符
set_tb_margin	***smgtb***	***MT***	设置上边距和下边距
set_top_margin	***smgt***	***Zo***	设置当前行的上边距
set_top_margin_parm	***smgtp***	***Zp***	设置第 #1 (#2) 行的上（下）边距
set_window	***wind***	***wi***	当前窗口是从第 #1 行到第 #2 行，从第 #3 列到第 #4 列
start_bit_image	***sbim***	***Zq***	开始打印位图图形
start_char_set_def	***scsd***	***Zr***	开始定义字符集
stop_bit_image	***rbim***	***Zs***	结束打印位图图形
stop_char_set_def	***rcsd***	***Zt***	结束定义字符集
subscript_characters	***subcs***	***Zu***	“支持下标的”字符的列表
superscript_characters	***supcs***	***Zv***	“支持上标的”字符的列表
tab	***ht***	***ta***	使用 Tab 键跳转到下一个占据 8 个空格的硬件制表位
these_cause_cr	***docr***	***Zw***	打印其中的任何字符将导致 **cr**
to_status_line	***tsl***	***ts***	转至状态行，第 #1 列
tone	***tone***	***TO***	选择音频拨号
user0	***u0***	***U0***	用户字符串 0
user1	***u1***	***U1***	用户字符串 1
user2	***u2***	***U2***	用户字符串 2
user3	***U3***	***u3***	用户字符串 3
user4	***u4***	***u4***	用户字符串 4
user5	***u5***	***u5***	用户字符串 5
user6	***u6***	***u6***	用户字符串 6
user7	***u7***	***u7***	用户字符串 7
user8	***u8***	***u8***	用户字符串 8
user9	***u9***	***u9***	用户字符串 9
underline_char	***uc***	***uc***	给某个字符划下划线并移过它
up_half_line	***hu***	***hu***	上移半行（反转 1/2 行）
wait_tone	***wait***	***WA***	等待拨号音
xoff_character	***xoffc***	***XF***	X-off 字符
xon_character	***xonc***	***XN***	X-on 字符
zero_motion	***zerom***	***Zx***	后续字符没有移动

以下内容可声明为函数并且可定义为宏：

int tgetent(char *bp, char *name);

```
int    tgetflag(char id[2]);
int    tgetnum(char id[2]);
char  *tgetstr(char id[2], char **area);
char  *tgoto(char *cap, int col, int row);
int    tputs(char *str, int affcnt, int (*putc)(void));
```

另请参阅

printf(1)。

名称

terminfo - 打印机、终端和调制解调器功能数据库

概要

/usr/lib/terminfo/?/*

说明

本联机帮助页中的要求仅对声明 Curses 增强的符合性的实现有效。

说明中各节标题的列表

Terminfo 源格式
源文件语法
最小保证限制
正式语法
定义的功能
示例条目
示例条目中的功能类型
设备功能
插入和（或）删除行
打印机功能
引发移动的功能
备用字符集
点阵图形
更改打印分辨率的效果
选择终端
应用程序用法

Terminfo 源格式

terminfo 数据库包含有关各种设备（如终端和打印机）功能的说明。设备通过指定功能集，将设备的某些方面量化以及指定将影响特定结果的字符序列来进行描述。

该联机帮助页指定 **terminfo** 源文件的格式。

符合 X/Open 的实现必须提供一个工具，它接受以本联机帮助页中指定的格式的源文件作为向 **terminfo** 数据库输入信息的方法。用于将该信息装入数据库的工具是针对于实现的。可以将描述给定终端型号的有效 **terminfo** 条目添加到任何符合 X/Open 的实现的 **terminfo**，以便使用相同的终端型号。

源文件语法一节介绍 **terminfo** 源文件的语法。语法和词汇惯例在下面的正式语法一节介绍。 X/Open 定义的所有终端功能的列表在下面的定义的功能一节提供。随后的示例条目一节将提供一个示例。设备功能一节将在总体上描述设备（如视频终端）的规范。打印机功能一节介绍打印机的规范。

terminfo 数据库通常由面向屏幕的应用程序（如 **vi** 和 Curses 程序）以及某些实用程序（如 **ls** 和 **more**）使用。通过使用该数据库，这些程序可以处理大量的设备，而无须更改程序。

源文件语法

源文件可以使用 ISO 8859-1 代码集。源文件使用其他代码集时的行为尚未确定。传统惯例是将信息从其他代码集转换为源文件语法。

terminfo 源文件由一个或多个设备说明组成。每个说明定义终端型号的助记名称。每个说明包括标头（在第 1 列中开始）和列出该特定设备功能的一个或多个行。 **terminfo** 源文件中的每一行必须以逗点结尾。 **terminfo** 源文件中除标头之外的每一行必须缩进一个或多个空白字符（空格或制表符）。

terminfo 源文件中的条目由大量逗点分隔的字段组成。将忽略每个逗点之后的空白字符。嵌入的逗点必须使用反斜杠转义。以下示例显示 **terminfo** 源文件的格式：

> *alias1* | *alias2* | ... | *aliasn* | *longname*,
> *whitespace* **am, lines #24,**
> *whitespace* **home=\Eeh,**

第一行通常称作标头行，它必须在第一列中开始，并且必须包含至少两个由竖线分隔的别名。标头行中的最后一个字段必须是设备的完整名称，它可以包含任意字符串。

别名在 **terminfo** 数据库中必须是唯一的，它们必须符合实现专用 **terminfo** 编译实用程序建立的文件命名惯例。实现将识别仅由可移植文件名字符集中的字符组成的别名，不过实现不需要接受第一个符号字符 **(-)**。例如，通常的限制是它们不能包含空白字符和斜线。实现专用的 **terminfo** 编译实用程序还可能对源文件值施加进一步的限制。

terminfo 中的每一功能属于以下类型之一：

- 布尔功能显示设备具有或不具有特定的功能。
- 数字功能将设备的特定功能量化。
- 字符串功能提供可用于在设备上执行特定功能的序列。

功能名称遵循五个字符长度的非正式限制。只要可能，请选择与 ANSI X3.64-1979 标准指定的名称相同或类似的功能名称。语义也应该匹配 ANSI 标准的语义。

除用于输入的字符串功能之外，所有字符串功能均可以指定填充。输入功能在下表的 **Strings** 部分列出，它们的名称以 **key_** 开头。这些功能在 **<term.h>** 中定义。

最小保证限制

对于 **terminfo** 源文件，所有符合 X/Open 的实现至少支持以下限制：

源文件特性	最小保证值
行长度	1023 字节
终端别名长度	14 字节
终端型号名称长度	128 字节
单个字段宽度	128 字节
字符串值长度	1000 字节
表示数字值的字符串的长度	99 位
数字值范围	0 到 32767（包括这两个值）

实现可能会支持高于上述限制的限制。

正式语法

本节中的语法和词汇惯例一起描述 **terminfo** 终端说明在 **terminfo** 源文件中的语法。所有实现均会接受满足本节要求的终端说明（记号“*(n)*”是指说明之后的注释）。

descriptions : START_OF_HEADER_LINE *(1)* **rest_of_header_line feature_lines**
  | descriptions START_OF_HEADER_LINE rest_of_header_line
  | feature_lines
  ;

rest_of_header_line : PIPE LONGNAME COMMA NEWLINE
  | aliases PIPE LONGNAME COMMA NEWLINE
  ;

feature_lines : start_feature_line rest_of_feature_line
  | feature_lines start_feature_line rest_of_feature_line
  ;

start_feature_line : START_FEATURE_LINE_BOOLEAN *(2)*
  | START_FEATURE_LINE_NUMERIC *(3)*
  | START_FEATURE_LINE_STRING *(4)*
  ;

rest_of_feature_line : features COMMA NEWLINE
  | COMMA NEWLINE
  ;

features : COMMA feature
  | features COMMA feature
  ;

aliases : PIPE ALIAS
| aliases PIPE ALIAS
;

feature : BOOLEAN
| NUMERIC
| STRING
;

Note *(1)* 在第一列中开始的别名。这由词汇分析程序来处理。

Note *(2)* 在第一列后开始但却是功能行上第一项功能的布尔功能。这由词汇分析程序来处理。

Note *(3)* 在第一列后开始但却是功能行上第一项功能的数字功能。这由词汇分析程序来处理。

Note *(4)* 在第一列后开始但却是功能行上第一项功能的字符串功能。这由词汇分析程序来处理。

terminfo 说明的词汇惯例如下:

1. 空白字符包括 <空格> 和 <制表符>。
2. ALIAS 可以包含除逗点 (**,**)、斜线 (**/**) 和竖线 (**|**) 之外的任何图形字符（图形字符是 **isgraph()** 针对字符返回的是非零值；请参阅 *ctype*(3C)）。
3. LONGNAME 可以包含除逗点 (**,**) 和竖线 (**|**) 之外的任意打印字符（打印字符是 **isprint()** 对其返回非零的字符；请参阅 *ctype*(3C)）。
4. 布尔功能可以包含除逗点 (**,**)、等号 (**=**) 和井号 (**#**) 之外的任何打印字符。
5. 数字功能包括:
   a. 名称，它可以包含除逗点 (**,**)、等号 (**=**) 和井号 (**#**) 之外的任何打印字符。
   b. 井号 (**#**) 字符。
   c. 正整数，它符合 C 语言的整数常量惯例。
6. 字符串功能包括:
   a. 名称，它可以包含除逗点 (**,**)、等号 (**=**) 和井号 (**#**) 之外的任何打印字符。
   b. 等号 (**=**) 字符。
   c. 字符串，它可以包含除逗点 (**,**) 之外的任何打印字符。
7. 将忽略紧接逗点 (**,**) 之后的空白字符。
8. 注释包括行开头、（可选）空白字符、（必需）井号 (**#**) 和行的终止。
9. 标头行必须在第一列中开始。

10. 功能行不得在第一列中开始。

11. 将忽略空白行。

定义的功能

X/Open 定义了下表所列的功能。所有符合 X/Open 的实现必须在 **terminfo** 源文件的条目中接受列出的各项功能。实现使用该信息确定如何正确地运行当前终端。此外，当应用程序调用 **tgetent()** （在下表列出 **Termcap** 代码的情况下）和 **tigetflag()** 中列出的查询函数时，实现将返回当前终端的任何功能（请参阅 *tgetent*(3X) 和 *tigetflag*(3X)）。

功能表包括以下列:

变量　　在 **terminfo** 数据库上执行操作的 Curses 函数所使用的名称。这些名称是保留字，应用程序不得定义它们。

功能名　　**terminfo** 源文件中指定的功能的简写名称。它用于更新源文件，并由 **tput** 命令使用（请参阅 *tput*(1)）。

功能码　　为兼容早期应用程序而提供的代码。这些代码是 **TO BE WITHDRAWN**。因此，并非所有功能名 (**Capname**) 都具有功能码 (**Termcap**)。

说明　　功能的简单摘要。

布尔值

变量	功能名称	终端功能代码	说明
auto_left_margin	***bw***	***bw***	**cub1** 从第 0 列回绕到最后一列
auto_right_margin	***am***	***am***	终端包含自动边距
back_color_erase	***bce***	***ut***	清除背景色的屏幕
can_change	***ccc***	***cc***	终端可重新定义现有颜色
ceol_standout_glitch	***xhp***	***xs***	未通过覆盖清除的标准输出 (hp)
col_addr_glitch	***xhpa***	***YA***	仅适用于 **hpa/mhpa** 功能的正向移动
cpi_changes_res	***cpix***	***YF***	更改字符间距将更改分辨率
cr_cancels_micro_mode	***crxm***	***YB***	使用 **cr** 关闭微观模式
dest_tabs_magic_smso	***xt***	***xt***	破坏性制表符，特别 **smso** 字符 (t1061)
eat_newline_glitch	***xenl***	***xn***	忽略 80 列后的换行符 (Concept)
erase_overstrike	***eo***	***eo***	可以用空白字符清除叠印
generic_type	***gn***	***gn***	一般行类型（如拨号、交换）
get_mouse	***getm***	***Gm***	Curses 应该获取按钮事件
hard_copy	***hc***	***hc***	硬拷贝终端
hard_cursor	***chts***	***HC***	很难看到光标

has_meta_key	*km*	*km*	包含转换键（Shift，设置奇偶校验位）
has_print_wheel	*daisy*	*YC*	打印机需要操作人员更改字符集
has_status_line	*hs*	*hs*	包含额外的"状态行"
hue_lightness_saturation	*hls*	*hl*	终端仅使用 HLS 颜色表示法（Tektronix）
insert_null_glitch	*in*	*in*	插入模式区分空值
lpi_changes_res	*lpix*	*YG*	更改行间距将更改分辨率
memory_above	*da*	*da*	显示可能保留在屏幕之上
memory_below	*db*	*db*	显示可能保留在屏幕之下
move_insert_mode	*mir*	*mi*	在插入模式下可以安全移动
move_standout_mode	*msgr*	*ms*	在标准输出模式下可以安全移动
needs_xon_xoff	*nxon*	*nx*	填充无效，需要 XON/XOFF
no_esc_ctlc	*xsb*	*xb*	Beehive（f1=escape、f2=ctrl C）
no_pad_char	*npc*	*NP*	填充字符不存在
non_dest_scroll_region	*ndscr*	*ND*	滚动区域不是破坏性的
non_rev_rmcup	*nrrmc*	*NR*	**smcup** 不反转 **rmcup**
over_strike	*os*	*os*	终端在硬拷贝终端上叠印
prtr_silent	*mc5i*	*5i*	打印机将不在屏幕上回显
row_addr_glitch	*xvpa*	*YD*	仅适用于 **vpa/mvpa** 功能的正向移动
semi_auto_right_margin	*sam*	*YE*	打印最后一列会导致 **cr**
status_line_esc_ok	*eslok*	*es*	可以在状态行上使用转义
tilde_glitch	*hz*	*hz*	Hazeltine；无法打印波浪符 (˜)
transparent_underline	*ul*	*ul*	给字符叠印添加下划线
xon_xoff	*xon*	*xo*	终端使用 XON/XOFF 握手

数字

变量	功能 名称	终端 功能代码	说明
bit_image_entwining	*bitwin*	*Yo*	每个位图行的扫描次数
bit_image_type	*bitype*	*Yp*	位图设备类型
buffer_capacity	*bufsz*	*Ya*	在打印前缓冲的字节数
buttons	*btns*	*BT*	鼠标上的按钮数
columns	*cols*	*co*	行中的列数
dot_horz_spacing	*spinh*	*Yc*	点水平间距（每英寸的点数）
dot_vert_spacing	*spinv*	*Yb*	针垂直间距（每英寸的针数）
init_tabs	*it*	*it*	最初每 **#** 个空格添加一个制表符
label_height	*lh*	*lh*	每个标签中的行数

label_width	***lw***	***lw***	每个标签中的列数
lines	***lines***	***li***	屏幕或页面上的行数
lines_of_memory	***lm***	***lm***	如果大于 **lines**，则表示内存行数；0 表示不定
max_attributes	***ma***	***ma***	终端可显示的最大组合视频属性
magic_cookie_glitch	***xmc***	***sg***	**smso** 或 **rmso** 留下的空白字符数
max_colors	***colors***	***Co***	屏幕上的最大颜色数
max_micro_address	***maddr***	***Yd***	**micro_..._address** 中的最大值
max_micro_jump	***mjump***	***Ye***	**parm_..._micro** 中的最大值
max_pairs	***pairs***	***pa***	屏幕上的最大颜色对数
maximum_windows	***wnum***	***MW***	最大的可定义窗口数
micro_col_size	***mcs***	***Yf***	处于微观模式时的字符步进大小
micro_line_size	***mls***	***Yg***	处于微观模式时的行步进大小
no_color_video	***ncv***	***NC***	无法与颜色一起使用的视频属性
num_labels	***nlab***	***Nl***	屏幕上的标签数（从 1 开始）
number_of_pins	***npins***	***Yh***	打印头中的针数
output_res_char	***orc***	***Yi***	水平分辨率（每个字符的单位数）
output_res_line	***orl***	***Yj***	垂直分辨率（每行的单位数）
output_res_horz_inch	***orhi***	***Yk***	水平分辨率（每英寸的单位数）
output_res_vert_inch	***orvi***	***Yl***	垂直分辨率（每英寸的单位数）
padding_baud_rate	***pb***	***pb***	需要填充时的最低波特率
print_rate	***cps***	***Ym***	打印速率（每秒的字符数）
virtual_terminal	***vt***	***vt***	虚拟终端号
wide_char_size	***widcs***	***Yn***	处于双倍宽度模式时的字符步进大小
width_status_line	***wsl***	***ws***	状态行中的列数

字符串（第 1 部分，共 3 个部分）

变量	功能 名称	终端 功能代码	说明
acs_chars	***acsc***	***ac***	图形字符集对 aAbBcC
alt_scancode_esc	***scesa***	***S8***	扫描代码模拟的备用转义（缺省值是 VT100）
back_tab	***cbt***	***bt***	向后切换
bell	***bel***	***bl***	听觉信号（响铃）
bit_image_carriage_return	***bicr***	***Yv***	移到同一行的开头
bit_image_newline	***binel***	***Zz***	移到位图的下一行
bit_image_repeat	***birep***	***Xy***	将位图单元格 #1 重复 #2 次

carriage_return	*cr*	*cr*	回车
change_char_pitch	*cpi*	*ZA*	更改每英寸的字符数
change_line_pitch	*lpi*	*ZB*	更改每英寸的行数
change_res_horz	*chr*	*ZC*	更改水平分辨率
change_res_vert	*cvr*	*ZD*	更改垂直分辨率
change_scroll_region	*csr*	*cs*	更改到行 #1 至行 #2 (VT100)
char_padding	*rmp*	*rP*	与 **ip** 类似，但在替换模式下
char_set_names	*csnm*	*Zy*	返回字符集名称列表
clear_all_tabs	*tbc*	*ct*	清除所有制表位
clear_margins	*mgc*	*MC*	清除所有边界（顶部、底部和两侧）
clear_screen	*clear*	*cl*	清除屏幕并将光标复位
clr_bol	*el1*	*cb*	清除到行开头（包括行开头）
clr_eol	*el*	*ce*	清除到行的结尾
clr_eos	*ed*	*cd*	清除到显示的结尾
code_set_init	*csin*	*ci*	多个代码集的初始序列
color_names	*colornm*	*Yw*	颜色 #1 的给定名称
column_address	*hpa*	*ch*	将水平位置设置为绝对 #1
command_character	*cmdch*	*CC*	原型中的终端可设置 cmd 字符
create_window	*cwin*	*CW*	定义窗口 #1 从 #2,#3 转到 #4,#5
cursor_address	*cup*	*cm*	移到行 #1 列 #2
cursor_down	*cud1*	*do*	下移一行
cursor_home	*home*	*ho*	将光标复位（如果无 **cup**）
cursor_invisible	*civis*	*vi*	使光标不可见
cursor_left	*cub1*	*le*	左移一个空格
cursor_mem_address	*mrcup*	*CM*	内存相对光标寻址
cursor_normal	*cnorm*	*ve*	使光标显得正常（撤消 **vs/vi**）
cursor_right	*cuf1*	*nd*	非破坏性空格（光标或回车向右）
cursor_to_ll	*ll*	*ll*	最后一行，第一列（如果无 **cup**）
cursor_up	*cuu1*	*up*	上移一行（光标上移）
cursor_visible	*cvvis*	*vs*	使光标非常可见
define_bit_image_region	*defbi*	*Yx*	定义矩形位图区
define_char	*defc*	*ZE*	定义字符集中的字符
delete_character	*dch1*	*dc*	删除字符
delete_line	*dl1*	*dl*	删除行
device_type	*devt*	*dv*	指示语言和（或）代码集支持
dial_phone	*dial*	*DI*	拨打电话号码 #1
dis_status_line	*dsl*	*ds*	禁用状态行

display_clock	***dclk***	***DK***	显示一天时间时钟
display_pc_char	***dispc***	***S1***	显示 PC 字符
down_half_line	***hd***	***hd***	下移半行（将换行符前移 1/2）
ena_acs	***enacs***	***eA***	启用备用字符集
end_bit_image_region	***endbi***	***Yy***	结束位图区
enter_alt_charset_mode	***smacs***	***as***	开始备用字符集
enter_am_mode	***smam***	***SA***	启用自动边距
enter_blink_mode	***blink***	***mb***	启用闪烁
enter_bold_mode	***bold***	***md***	启用粗体（格外明亮）模式
enter_ca_mode	***smcup***	***ti***	启动使用 **cup** 的程序的字符串
enter_delete_mode	***smdc***	***dm***	删除模式 (enter)
enter_dim_mode	***dim***	***mh***	启用半明亮模式
enter_doublewide_mode	***swidm***	***ZF***	启用双倍宽度打印
enter_draft_quality	***sdrfq***	***ZG***	设置草稿品质打印
enter_horizontal_hl_mode	***ehhlm***		启用水平突出显示模式
enter_insert_mode	***smir***	***im***	插入模式 (enter)
enter_italics_mode	***sitm***	***ZH***	启用斜体
enter_left_hl_mode	***elhlm***		启用左侧突出显示模式
enter_leftward_mode	***slm***	***ZI***	启用向左回车移动
enter_low_hl_mode	***elohlm***		启用底部突出显示模式
enter_micro_mode	***smicm***	***ZJ***	启用微观移动功能
enter_near_letter_quality	***snlq***	***ZK***	设置仿信函品质打印
enter_normal_quality	***snrmq***	***ZL***	设置常规品质打印
enter_pc_charset_mode	***smpch***	***S2***	进入 PC 字符显示模式
enter_protected_mode	***prot***	***mp***	启用保护模式
enter_reverse_mode	***rev***	***mr***	启用反白显示模式
enter_right_hl_mode	***erhlm***		启用右侧突出显示模式
enter_scancode_mode	***smsc***	***S4***	进入 PC 扫描代码模式
enter_secure_mode	***invis***	***mk***	启用空白模式（字符不可见）
enter_shadow_mode	***sshm***	***ZM***	启用阴影打印
enter_standout_mode	***smso***	***so***	开始标准输出模式
enter_subscript_mode	***ssubm***	***ZN***	启用下标打印
enter_superscript_mode	***ssupm***	***ZO***	启用上标打印
enter_top_hl_mode	***ethlm***		启用顶部突出显示模式
enter_underline_mode	***smul***	***us***	开始下划线模式
enter_upward_mode	***sum***	***ZP***	启用向上回车移动
enter_vertical_hl_mode	***evhlm***		启用垂直突出显示模式
enter_xon_mode	***smxon***	***SX***	启用 XON/XOFF 握手

erase_chars	*ech*	*ec*	清除 #1 个字符
exit_alt_charset_mode	*rmacs*	*ae*	结束备用字符集
exit_am_mode	*rmam*	*RA*	禁用自动边距
exit_attribute_mode	*sgr0*	*me*	禁用所有属性
exit_ca_mode	*rmcup*	*te*	结束使用 **cup** 的程序的字符串
exit_delete_mode	*rmdc*	*ed*	结束删除模式
exit_doublewide_mode	*rwidm*	*ZQ*	禁用双倍宽度打印
exit_insert_mode	*rmir*	*ei*	结束插入模式
exit_italics_mode	*ritm*	*ZR*	禁用斜体
exit_leftward_mode	*rlm*	*ZS*	启用向右（常规）回车移动
exit_micro_mode	*rmicm*	*ZT*	禁用微观移动功能
exit_pc_charset_mode	*rmpch*	*S3*	禁用 PC 字符显示模式
exit_scancode_mode	*rmsc*	*S5*	禁用 PC 扫描代码模式
exit_shadow_mode	*rshm*	*ZU*	禁用阴影打印
exit_standout_mode	*rmso*	*se*	结束标准输出模式
exit_subscript_mode	*rsubm*	*ZV*	禁用下标打印
exit_superscript_mode	*rsupm*	*ZW*	禁用上标打印
exit_underline_mode	*rmul*	*ue*	结束下划线模式
exit_upward_mode	*rum*	*ZX*	启用向下（常规）回车移动
exit_xon_mode	*rmxon*	*RX*	禁用 XON/XOFF 握手
fixed_pause	*pause*	*PA*	暂停 2–3 秒钟
flash_hook	*hook*	*fh*	使开关钩闪烁
flash_screen	*flash*	*vb*	可见响铃（可以移动鼠标）
form_feed	*ff*	*ff*	硬拷贝终端页弹出
from_status_line	*fsl*	*fs*	从状态行返回
goto_window	*wingo*	*WG*	转到窗口 #1
hangup	*hup*	*HU*	挂断电话
init_1string	*is1*	*i1*	终端或打印机初始化字符串
init_2string	*is2*	*is*	终端或打印机初始化字符串
init_3string	*is3*	*i3*	终端或打印机初始化字符串
init_file	*if*	*if*	初始化文件名
init_prog	*iprog*	*iP*	初始化程序路径名
initialize_color	*initc*	*IC*	将颜色 #1 设置为 RGB #2、#3、#4
initialize_pair	*initp*	*Ip*	将颜色对 #1 设置为 RGB #2、#3、#4 (fg) 和 RGB #5、#6、#7 (bg)
insert_character	*ich1*	*ic*	插入字符
insert_line	*il1*	*al*	添加新的空白行
insert_padding	*ip*	*ip*	在插入的字符后插入填充

字符串（第 2 部分，共 3 个部分）

“**key_**”字符串通过特定键发送。“**key_**”说明包括在按键时 **getch()** 为代码返回的宏（在 **<curses.h>** 中定义）（请参阅 *getch*(3X)）。

变量	功能 名称	终端 功能代码	说明
key_a1	***ka1***	***K1***	键盘的左上部分
key_a3	***ka3***	***K3***	键盘的右上部分
key_b2	***kb2***	***K2***	键盘的中心部分
key_backspace	***kbs***	***kb***	通过退格键发送
key_beg	***kbeg***	***@1***	通过 beg（开始）键发送
key_btab	***kcbt***	***kB***	通过 back-tab 键发送
key_c1	***kc1***	***K4***	键盘的左下部分
key_c3	***kc3***	***K5***	键盘的右下部分
key_cancel	***kcan***	***@2***	通过 Cancel 键发送
key_catab	***ktbc***	***ka***	通过 clear-all-tabs 键发送
key_clear	***kclr***	***kC***	通过 clear-screen 或 erase 键发送
key_close	***kclo***	***@3***	通过 close 键发送
key_command	***kcmd***	***@4***	通过 cmd（命令）键发送
key_copy	***kcpy***	***@5***	通过 Copy 键发送
key_create	***kcrt***	***@6***	通过 Create 键发送
key_ctab	***kctab***	***kt***	通过 clear-tab 键发送
key_dc	***kdch1***	***kD***	通过 delete-character 键发送
key_dl	***kdl1***	***kL***	通过 delete-line 键发送
key_down	***kcud1***	***kd***	通过终端向下箭头键发送
key_eic	***krmir***	***kM***	在插入模式下通过 **rmir** 或 **smir** 发送
key_end	***kend***	***@7***	通过 end 键发送
key_enter	***kent***	***@8***	通过 enter 和（或）send 键发送
key_eol	***kel***	***kE***	通过 clear-to-end-of-line 键发送
key_eos	***ked***	***kS***	通过 clear-to-end-of-screen 键发送
key_exit	***kext***	***@9***	通过 exit 键发送
key_f0	***kf0***	***k0***	通过功能键 f0 发送
key_f1	***kf1***	***k1***	通过功能键 f1 发送
key_f2	***kf2***	***k2***	通过功能键 f2 发送
key_f3	***kf3***	***k3***	通过功能键 f3 发送
key_f4	***kf4***	***k4***	通过功能键 f4 发送
key_f5	***kf5***	***k5***	通过功能键 f5 发送
key_f6	***kf6***	***k6***	通过功能键 f6 发送

key_f7	***kf7***	***k7***	通过功能键 f7 发送
key_f8	***kf8***	***k8***	通过功能键 f8 发送
key_f9	***kf9***	***k9***	通过功能键 f9 发送
key_f10	***kf10***	***k;***	通过功能键 f10 发送
key_f11	***kf11***	***F1***	通过功能键 f11 发送
key_f12	***kf12***	***F2***	通过功能键 f12 发送
key_f13	***kf13***	***F3***	通过功能键 f13 发送
key_f14	***kf14***	***F4***	通过功能键 f14 发送
key_f15	***kf15***	***F5***	通过功能键 f15 发送
key_f16	***kf16***	***F6***	通过功能键 f16 发送
key_f17	***kf17***	***F7***	通过功能键 f17 发送
key_f18	***kf18***	***F8***	通过功能键 f18 发送
key_f19	***kf19***	***F9***	通过功能键 f19 发送
key_f20	***kf20***	***FA***	通过功能键 f20 发送
key_f21	***kf21***	***FB***	通过功能键 f21 发送
key_f22	***kf22***	***FC***	通过功能键 f22 发送
key_f23	***kf23***	***FD***	通过功能键 f23 发送
key_f24	***kf24***	***FE***	通过功能键 f24 发送
key_f25	***kf25***	***FF***	通过功能键 f25 发送
key_f26	***kf26***	***FG***	通过功能键 f26 发送
key_f27	***kf27***	***FH***	通过功能键 f27 发送
key_f28	***kf28***	***FI***	通过功能键 f28 发送
key_f29	***kf29***	***FJ***	通过功能键 f29 发送
key_f30	***kf30***	***FK***	通过功能键 f30 发送
key_f31	***kf31***	***FL***	通过功能键 f31 发送
key_f32	***kf32***	***FM***	通过功能键 f32 发送
key_f33	***kf33***	***FN***	通过功能键 f33 发送
key_f34	***kf34***	***FO***	通过功能键 f34 发送
key_f35	***kf35***	***FP***	通过功能键 f35 发送
key_f36	***kf36***	***FQ***	通过功能键 f36 发送
key_f37	***kf37***	***FR***	通过功能键 f37 发送
key_f38	***kf38***	***FS***	通过功能键 f38 发送
key_f39	***kf39***	***FT***	通过功能键 f39 发送
key_f40	***kf40***	***FU***	通过功能键 f40 发送
key_f41	***kf41***	***FV***	通过功能键 f41 发送
key_f42	***kf42***	***FW***	通过功能键 f42 发送
key_f43	***kf43***	***FX***	通过功能键 f43 发送
key_f44	***kf44***	***FY***	通过功能键 f44 发送

key_f45	***kf45***	***FZ***	通过功能键 f45 发送
key_f46	***kf46***	***Fa***	通过功能键 f46 发送
key_f47	***kf47***	***Fb***	通过功能键 f47 发送
key_f48	***kf48***	***Fc***	通过功能键 f48 发送
key_f49	***kf49***	***Fd***	通过功能键 f49 发送
key_f50	***kf50***	***Fe***	通过功能键 f50 发送
key_f51	***kf51***	***Ff***	通过功能键 f51 发送
key_f52	***kf52***	***Fg***	通过功能键 f52 发送
key_f53	***kf53***	***Fh***	通过功能键 f53 发送
key_f54	***kf54***	***Fi***	通过功能键 f54 发送
key_f55	***kf55***	***Fj***	通过功能键 f55 发送
key_f56	***kf56***	***Fk***	通过功能键 f56 发送
key_f57	***kf57***	***Fl***	通过功能键 f57 发送
key_f58	***kf58***	***Fm***	通过功能键 f58 发送
key_f59	***kf59***	***Fn***	通过功能键 f59 发送
key_f60	***kf60***	***Fo***	通过功能键 f60 发送
key_f61	***kf61***	***Fp***	通过功能键 f61 发送
key_f62	***kf62***	***Fq***	通过功能键 f62 发送
key_f63	***kf63***	***Fr***	通过功能键 f63 发送
key_find	***kfnd***	***@0***	通过 find 键发送
key_help	***khlp***	***%1***	通过 help 键发送
key_home	***khome***	***kh***	通过 home 键发送
key_ic	***kich1***	***kI***	通过 ins-char 和（或）enter ins-mode 键发送
key_il	***kil1***	***kA***	通过 insert-line 键发送
key_left	***kcub1***	***kl***	通过终端向左箭头键发送
key_ll	***kll***	***kH***	通过 home-down 键发送
key_mark	***kmrk***	***%2***	通过 mark 键发送
key_message	***kmsg***	***%3***	通过 message 键发送
key_mouse	***kmous***	***Km***	0631，鼠标事件已发生
key_move	***kmov***	***%4***	通过 move 键发送
key_next	***knxt***	***%5***	通过 next-object 键发送
key_npage	***knp***	***kN***	通过 next-page 键发送
key_open	***kopn***	***%6***	通过 open 键发送
key_options	***kopt***	***%7***	通过 options 键发送
key_ppage	***kpp***	***kP***	通过 previous-page 键发送
key_previous	***kprv***	***%8***	通过 previous-object 键发送
key_print	***kprt***	***%9***	通过 print 或 copy 键发送
key_redo	***krdo***	***%0***	通过 redo 键发送

key_reference	***kref***	***&1***	通过 ref（参考）键发送
key_refresh	***krfr***	***&2***	通过 refresh 键发送
key_replace	***krpl***	***&3***	通过 replace 键发送
key_restart	***krst***	***&4***	通过 restart 键发送
key_resume	***kres***	***&5***	通过 resume 键发送
key_right	***kcuf1***	***kr***	通过终端向右箭头键发送
key_save	***ksav***	***&6***	通过 save 键发送
key_sbeg	***kBEG***	***&9***	通过切换的开始键发送
key_scancel	***kCAN***	***&0***	通过切换的 cancel 键发送
key_scommand	***kCMD***	****1***	通过切换的命令键发送
key_scopy	***kCPY***	****2***	通过切换的 copy 键发送
key_screate	***kCRT***	****3***	通过切换的 create 键发送
key_sdc	***kDC***	****4***	通过切换的 delete-char 键发送
key_sdl	***kDL***	****5***	通过切换的 delete-line 键发送
key_select	***kslt***	****6***	通过 select 键发送
key_send	***kEND***	****7***	通过切换的 end 键发送
key_seol	***kEOL***	****8***	通过切换的 clear-line 键发送
key_sexit	***kEXT***	****9***	通过切换的 exit 键发送
key_sf	***kind***	***kF***	通过向前和（或）向下滚动键发送
key_sfind	***kFND***	****0***	通过切换的 find 键发送
key_shelp	***kHLP***	***#1***	通过切换的 help 键发送
key_shome	***kHOM***	***#2***	通过切换的 home 键发送
key_sic	***kIC***	***#3***	通过切换的 input 键发送
key_sleft	***kLFT***	***#4***	通过切换的向左箭头键发送
key_smessage	***kMSG***	***%a***	通过切换的 message 键发送
key_smove	***kMOV***	***%b***	通过切换的 move 键发送
key_snext	***kNXT***	***%c***	通过切换的 next 键发送
key_soptions	***kOPT***	***%d***	通过切换的 options 键发送
key_sprevious	***kPRV***	***%e***	通过切换的 prev 键发送
key_sprint	***kPRT***	***%f***	通过切换的 print 键发送
key_sr	***kri***	***kR***	通过向后和（或）向上滚动键发送
key_sredo	***kRDO***	***%g***	通过切换的 redo 键发送
key_sreplace	***kRPL***	***%h***	通过切换的 replace 键发送
key_sright	***kRIT***	***%i***	通过切换的向右箭头键发送
key_srsume	***kRES***	***%j***	通过切换的 resume 键发送
key_ssave	***kSAV***	***!1***	通过切换的 save 键发送
key_ssuspend	***kSPD***	***!2***	通过切换的 suspend 键发送
key_stab	***khts***	***kT***	通过 set-tab 键发送

key_sundo	***kUND***	***!3***	通过切换的 undo 键发送
key_suspend	***kspd***	***&7***	通过 suspend 键发送
key_undo	***kund***	***&8***	通过 undo 键发送
key_up	***kcuu1***	***ku***	通过终端向上箭头键发送

字符串（第 3 部分，共 3 个部分）

变量	功能名称	终端功能代码	说明
keypad_local	***rmkx***	***ke***	退出“键盘传输”模式
keypad_xmit	***smkx***	***ks***	将终端置于“键盘传输”模式
lab_f0	***lf0***	***l0***	功能键 f0 上的标签（如果不是 f0）
lab_f1	***lf1***	***l1***	功能键 f1 上的标签（如果不是 f1）
lab_f2	***lf2***	***l2***	功能键 f2 上的标签（如果不是 f2）
lab_f3	***lf3***	***l3***	功能键 f3 上的标签（如果不是 f3）
lab_f4	***lf4***	***l4***	功能键 f4 上的标签（如果不是 f4）
lab_f5	***lf5***	***l5***	功能键 f5 上的标签（如果不是 f5）
lab_f6	***lf6***	***l6***	功能键 f6 上的标签（如果不是 f6）
lab_f7	***lf7***	***l7***	功能键 f7 上的标签（如果不是 f7）
lab_f8	***lf8***	***l8***	功能键 f8 上的标签（如果不是 f8）
lab_f9	***lf9***	***l9***	功能键 f9 上的标签（如果不是 f9）
lab_f10	***lf10***	***la***	功能键 f10 上的标签（如果不是 f10）
label_format	***fln***	***Lf***	标签格式
label_off	***rmln***	***LF***	禁用软标签
label_on	***smln***	***LO***	启用软标签
memory_lock	***meml***	***ml***	锁定光标之上的内存
memory_unlock	***memu***	***mu***	禁用内存锁
meta_off	***rmm***	***mo***	禁用“元模式”
meta_on	***smm***	***mm***	启用“元模式”（第 8 位）
micro_column_address	***mhpa***	***ZY***	对于微观调整与 **column_address** 类似
micro_down	***mcud1***	***ZZ***	对于微观调整与 **cursor_down** 类似
micro_left	***mcub1***	***Za***	对于微观调整与 **cursor_left** 类似
micro_right	***mcuf1***	***Zb***	对于微观调整与 **cursor_right** 类似
micro_row_address	***mvpa***	***Zc***	对于微观调整与 **row_address** 类似
micro_up	***mcuu1***	***Zd***	对于微观调整与 **cursor_up** 类似
mouse_info	***minfo***	***Mi***	鼠标状态信息
newline	***nel***	***nw***	换行符（类似于 **cr** 后接 **lf**）
order_of_pins	***porder***	***Ze***	将软件位与打印头的针匹配

orig_colors	***oc***	***oc***	将所有颜色（对）设置为原色（对）
orig_pair	***op***	***op***	将缺省颜色对设置为原色对
pad_char	***pad***	***pc***	填充字符（而不是空）
parm_dch	***dch***	***DC***	删除 #1 个字符
parm_delete_line	***dl***	***DL***	删除 #1 行
parm_down_cursor	***cud***	***DO***	下移 #1 行
parm_down_micro	***mcud***	***Zf***	对于微观调整与 **parm_down_cursor** 类似
parm_ich	***ich***	***IC***	插入 #1 个空白字符
parm_index	***indn***	***SF***	向前滚动 #1 行
parm_insert_line	***il***	***AL***	添加 #1 个新空白行
parm_left_cursor	***cub***	***LE***	将光标左移 #1 个空格
parm_left_micro	***mcub***	***Zg***	对于微观调整与 **parm_left_cursor** 类似
parm_right_cursor	***cuf***	***RI***	右移 #1 个空格
parm_right_micro	***mcuf***	***Zh***	对于微观调整与 **parm_right_cursor** 类似
parm_rindex	***rin***	***SR***	向后滚动 #1 行
parm_up_cursor	***cuu***	***UP***	将光标上移 #1 行
parm_up_micro	***mcuu***	***Zi***	对于微观调整与 **parm_up_cursor** 类似
pc_term_options	***pctrm***	***S6***	PC 终端选项
pkey_key	***pfkey***	***pk***	将功能键 #1 编程以确定字符串 #2 的类型
pkey_local	***pfloc***	***pl***	将功能键 #1 编程以执行字符串 #2
pkey_plab	***pfxl***	***xl***	将键 #1 编程以传输字符串 #2 并显示字符串 #3
pkey_xmit	***pfx***	***px***	将功能键 #1 编程以传输字符串 #2
plab_norm	***pln***	***pn***	将标签 #1 编程以显示字符串 #2
print_screen	***mc0***	***ps***	打印屏幕内容
prtr_non	***mc5p***	***pO***	为 #1 字节打开打印机
prtr_off	***mc4***	***pf***	关闭打印机
prtr_on	***mc5***	***po***	打开打印机
pulse	***pulse***	***PU***	选择脉冲拨号
quick_dial	***qdial***	***QD***	拨打电话号码 #1 而不进行进度检测
remove_clock	***rmclk***	***RC***	删除一天时间时钟
repeat_char	***rep***	***rp***	将字符 #1 重复 #2 次
req_for_input	***rfi***	***RF***	发送下一输入字符（为 ptys）
req_mouse_pos	***reqmp***	***RQ***	请求鼠标位置报告
reset_1string	***rs1***	***r1***	将终端完全重置为健全模式
reset_2string	***rs2***	***r2***	将终端完全重置为健全模式
reset_3string	***rs3***	***r3***	将终端完全重置为健全模式
reset_file	***rf***	***rf***	包含重置字符串的文件的名称

restore_cursor	***rc***	***rc***	将光标恢复到最后一个 sc 的位置
row_address	***vpa***	***cv***	将垂直位置设置为绝对 #1
save_cursor	***sc***	***sc***	保存光标位置
scancode_escape	***scesc***	***S7***	扫描代码模拟的转义
scroll_forward	***ind***	***sf***	向上滚动文本
scroll_reverse	***ri***	***sr***	向下滚动文本
select_char_set	***scs***	***Zj***	选择字符集
set0_des_seq	***s0ds***	***s0***	切换到字符集 0（EUC 字符集 0，ASCII）
set1_des_seq	***s1ds***	***s1***	切换到字符集 1
set2_des_seq	***s2ds***	***s2***	切换到字符集 2
set3_des_seq	***s3ds***	***s3***	切换到字符集 3
set_a_attributes	***sgr1***		定义第二组视频属性 #1–#6
set_a_background	***setab***	***AB***	使用 ANSI 转义将背景色设置为 #1
set_a_foreground	***setaf***	***AF***	使用 ANSI 转义将前景色设置为 #1
set_attributes	***sgr***	***sa***	定义第一组视频属性 #1–#9
set_background	***setb***	***Sb***	将背景色设置为 #1
set_bottom_margin	***smgb***	***Zk***	在当前行处设置下边距
set_bottom_margin_parm	***smgbp***	***Zl***	在离底部 #1 或 #2 行处设置下边距
set_clock	***sclk***	***SC***	将时钟设置为小时 (#1)、分钟 (#2)、秒钟 (#3)
set_color_band	***setcolor***	***Yz***	更改为条带颜色 #1
set_color_pair	***scp***	***sp***	将当前颜色对设置为 #1
set_foreground	***setf***	***Sf***	将前景色设置为 #1
set_left_margin	***smgl***	***ML***	在当前列处设置左边距
set_left_margin_parm	***smglp***	***Zm***	在列 #1 处设置左（右）边距 (#2)
set_lr_margin	***smglr***	***ML***	设置左右边距
set_page_length	***slines***	***YZ***	将页面长度设置为 #1 行
set_pglen_inch	***slength***	***YI***	将页面长度设置为一英寸的百分之 #1
set_right_margin	***smgr***	***MR***	在当前列处设置右边距
set_right_margin_parm	***smgrp***	***Zn***	在列 #1 处设置右边距
set_tab	***hts***	***st***	在所有行的当前列中设置制表符
set_tb_margin	***smgtb***	***MT***	设置上下边距
set_top_margin	***smgt***	***Zo***	在当前行处设置顶部边距
set_top_margin_parm	***smgtp***	***Zp***	在行 #1 处设置上（下）边距 (#2)
set_window	***wind***	***wi***	当前窗口是行 #1–#2 列 #3–#4
start_bit_image	***sbim***	***Zq***	开始打印位图图形
start_char_set_def	***scsd***	***Zr***	开始字符集定义
stop_bit_image	***rbim***	***Zs***	停止打印位图图形

stop_char_set_def	*rcsd*	*Zt*	结束字符集定义
subscript_characters	*subcs*	*Zu*	“可作下标”字符的列表
superscript_characters	*supcs*	*Zv*	“可作上标”字符的列表
tab	*ht*	*ta*	切换到下一个 8 空格硬件制表位
these_cause_cr	*docr*	*Zw*	打印其中任意字符将导致 **cr**
to_status_line	*tsl*	*ts*	转到状态行，列 #1
tone	*tone*	*TO*	选择按键式拨号
user0	*u0*	*u0*	用户字符串 0
user1	*u1*	*u1*	用户字符串 1
user2	*u2*	*u2*	用户字符串 2
user3	*u3*	*u3*	用户字符串 3
user4	*u4*	*u4*	用户字符串 4
user5	*u5*	*u5*	用户字符串 5
user6	*u6*	*u6*	用户字符串 6
user7	*u7*	*u7*	用户字符串 7
user8	*u8*	*u8*	用户字符串 8
user9	*u9*	*u9*	用户字符串 9
underline_char	*uc*	*uc*	给一个字符添加下划线并且经过它移动
up_half_line	*hu*	*hu*	上移半行（将换行符后移 1/2）
wait_tone	*wait*	*WA*	等待拨号音
xoff_character	*xoffc*	*XF*	XOFF 字符
xon_character	*xonc*	*XN*	XON 字符
zero_motion	*zerom*	*Zx*	对于后继字符没有移动

示例条目

以下条目描述 AT&T 610 终端（**pfxl** 和 **sgr** 值已经为打印而拆分；它们实际上将作为单个行输入）。

```
610|610bct|ATT610|att610|AT&T610;80column;98key keyboard,
    am, eslok, hs, mir, msgr, xenl, xon,
    cols#80, it#8, lh#2, lines#24, lw#8, nlab#8, wsl#80,
    acsc=“aaffggjjkkllmmnnooppqqrrssttuuvvwwxxyyzz{{||}}˜˜,
    bel=ˆG, blink=\E[5m, bold=\E[1m, cbt=\E[Z,
    civis=\E[?25l, clear=\E[H\E[J, cnorm=\E[?25h\E[?12l,
    cr=\r, csr=\E[%i%p1%d;%p2%dr, cub=\E[%p1%dD, cub1=\b,
    cud=\E[%p1%dB, cud1=\E[B, cuf=\E[%p1%dC, cuf1=\E[C,
    cup=\E[%i%p1%d;%p2%dH, cuu=\E[%p1%dA, cuu1=\E[A,
    cvvis=\E[?12;25h, dch=\E[%p1%dP, dch1=\E[P, dim=\E[2m,
    dl=\E[%p1%dM, dl1=\E[M, ed=\E[J, el=\E[K, el1=\E[1K,
    flash=\E[?5h$<200>\E[?5l, fsl=\E8, home=\E[H, ht=\t,
    ich=\E[%p1%d@, il=\E[%p1%dL, il1=\E[L, ind=\ED, .ind=\ED$<9>,
```

```
invis=\E[8m,
is1=\E[8;0 | \E[?3;4;5;13;15l\E[13;20l\E[?7h\E[12h\E(B\E)0,
is2=\E[0m^O, is3=\E(B\E)0, kLFT=\E[\s@, kRIT=\E[\sA,
kbs=^H, kcbt=\E[Z, kclr=\E[2J, kcub1=\E[D, kcud1=\E[B,
kcuf1=\E[C, kcuu1=\E[A, kfP=\EOc, kfP0=\ENp,
kfP1=\ENq, kfP2=\ENr, kfP3=\ENs, kfP4=\ENt, kfI=\EOd,
kfB=\EOe, kf4=\EOf, kf(CW=\EOg, kf6=\EOh, kf7=\EOi,
kf8=\EOj, kf9=\ENo, khome=\E[H, kind=\E[S, kri=\E[T,
ll=\E[24H, mc4=\E[?4i, mc5=\E[?5i, nel=\EE,
pfxl=\E[%p1%d;%p2%l%02dq%?%p1%{9}%<%t\s\s\sF%p1%1d
\s\s\s\s\s\s\s\s\s\s\s%;%p2%s,
pln=\E[%p1%d;0;0;0q%p2%:-16.16s, rc=\E8, rev=\E[7m,
ri=\EM, rmacs=^O, rmir=\E[4l, rmln=\E[2p, rmso=\E[m,
rmul=\E[m, rs2=\Ec\E[?3l, sc=\E7,
sgr=\E[0%?%p6%t;1%;%?%p5%t;2%;%?%p2%t;4%;%?%p4%t;5%;%?%p3%p1%
|%t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;,
sgr0=\E[m^O, smacs=^N, smir=\E[4h, smln=\E[p,
smso=\E[7m, smul=\E[4m, tsl=\E7\E[25;%i%p1%dx,
```

示例条目中的功能类型

该示例条目显示三种 **terminfo** 功能的格式：布尔、数字和字符串。在 **terminfo** 源文件中指定的所有功能后必须接有逗点，其中包括源文件中的最后一项功能。在 **terminfo** 源文件中，功能按其功能名称引用（如上表的 功能名列所示）。

布尔功能

如果 功能名存在于条目中，布尔功能为 true，如果 功能名不在条目中，则为 false。

功能名后的“@”字符用于显式声明在下文插入和（或）删除行一节的类似的终端小节中所述的情况下，布尔功能将是 false。

数字功能

数字功能后接“**#**”，其后是一个正整数值。该示例将值 80 赋给 **cols** 数字功能，其代码是：

cols#80

数字功能的值可以使用常规 C 语言惯例以十进制、八进制或十六进制形式指定。

字符串功能

字符串值功能（如 **el**（清除到行序列结尾）通过功能名 (**Capname**)、“**=**”和以下一个逗点实例结尾的字符串列出。

以毫秒为单位的延迟可能出现在这种功能中的任意位置，它前面加有“**$**”，并且包括在尖括号中，如 **el=\EK$<3>** 所示。 Curses 实现通过向终端输出适当数量的系统定义填充字符来实现延迟。 **tputs()** 函数提供在向终端发送这种功能时的延迟。

延迟可以是下面的任一种形式：数字；后接星号的数字，如 **5***；后接斜线的数字，如 **5/**；或者后接星号和斜线的数字，如 **5*/**。

***** 显示所需的延迟与操作所影响的行数成正比，并且给定的数量为每个受影响的设备所需的延迟（对于插入字符，该系数将仍然是受影响的行数。除非设备包含 **in** 并软件将使用它，否则它始终是 1）。当指定“*****”时，提供 **3.5** 形式的延迟来指定每个设备的延迟（单位是毫秒的十分之几）有时非常有用（只允许有一个小数位）。

/ 指示延迟是必需的，并且将传输填充字符，而不管 **xon** 的设置是什么。如果未指定“**/**”或者设备已定义 **xon** ，则延迟信息是建议性的，仅用于成本估算，或者用于设备处于原始模式的情况。但是，为 **bel** 或 **flash** 指定的任何延迟均被当作是必需的。

以下记号仅在 **terminfo** 源文件中才有效，它们用于指定特殊字符：

记号	表示字符
^*x*	Control-*x*（对于任何适当的 *x*）
a	警报
b	退格
E 或 **\e**	转义字符
f	换页符
l	换行符
n	换行符
r	回车
s	空格
t	制表符
\\^	脱字符 (**^**)
****	反斜杠 (****)
\\,	逗点 (**,**)
\\:	冒号 (**:**)
0	空
*****nnn*	任意字符，以三个八进制位的形式指定

（请参阅“X/Open 系统接口定义第 4 期第 2 版”规范“通用终端接口”）。

注释掉的功能

有时必须注释掉单独的功能。为此，在该功能名前添加一个句点。例如，请参阅上面示例条目一节中示例的 **ind**。请注意，功能按照从左到右的顺序定义，因此前面的定义将覆盖后面的定义。

设备功能

基本功能

设备在每一行上的列数由 **cols** 数字功能给定。如果设备带有屏幕，则屏幕上的行数由 **lines** 功能给定。如果设备在到达左边距时换行到下一行的开头，它应该具有 **am** 功能。如果终端可以清除其屏幕，并使光标复位，则屏幕上的行数由 **clear** 字符串功能给定。如果终端叠印（而不是在叠印字符时清除位置），则应该具有 **os** 功能。如果

设备是打印终端，且不带有软拷贝装置，请指定 **hc** 和 **os**。如果可以将光标移到当前行的左边缘，请将其指定为 **cr** （它通常是回车，control-M）。如果可以生成听觉信号（如铃声或嘟嘟声），请将其指定为 **bel**。如果与大多数设备一样，该设备使用 XON/XOFF 流控制协议，请指定 **xon**。

如果可以将光标向左移动一个位置（例如使用退格键），该功能应该指定为 **cub1**。同样，向右、向上和向下移动的序列应该分别指定为 **cuf1**、 **cuu1** 和 **cud1**。这些局部光标移动不得更改它们经过的文本；例如，通常不会使用"**cuf1=\s**"，因为空格会清除它经过的字符。

在这里，非常重要的一点是在 **terminfo** 中编码的局部光标移动在屏幕终端的左侧和顶部边缘是未定义的。除非指定了 **bw**，否则程序决不应该尝试在左边缘周围使用退格，并且决不应该尝试局部向上移动超过屏幕顶部。要向上滚动文本，程序将转到屏幕的左下角，并发送 **ind**（索引）字符串。要向下滚动文本，程序将转到屏幕的左上角，并发送 **ri**（反向索引）字符串。字符串 **ind** 和 **ri** 在不位于其对应的屏幕角落时是未定义的。

滚动序列的参数化版本是 **indn** 和 **rin**。除了采用一个参数并滚动该参数定义的行数之外，这些版本与 **ind** 和 **ri** 具有相同的语义。除了位于屏幕的适当边缘之外，它们也是未定义的。

am 功能指示光标是否在输出文本时停留在屏幕的右边缘，但它不一定适用于最后一列中的 **cuf1**。只有在指定 **bw** 时，才能从屏幕的左边缘向后移动。这种情况下，**cub1** 将移到上一行的右边缘。如果未给定 **bw**，其作用则是未定义的。例如，它可用于在屏幕边缘绘制方框。如果设备具有自动选择边距（通过参数）的功能，**am** 则应在 **terminfo** 源文件中指定。这种情况下，初始化字符串应该启用该选项（如果可能）。如果设备包含移到下一行第一列的命令，该命令可以给定为 **nel**（换行符）。该命令是否清除当前行的剩余部分并不重要，因此如果设备没有 **cr** 和 **lf**，仍可以通过其中的一个或两个命令创建有效的 **nel** 。

这些功能足以描述硬拷贝和屏幕终端。因此，AT&T 5320 硬拷贝终端的描述如下：

```
5320|att5320|AT&T 5320 hardcopy terminal,
    am, hc, os,
    cols#132,
    bel=^G, cr=\r, cub1=\b, cnd1=\n,
    dch1=\E[P, dl1=\E[M,
    ind=\n,
```

而 Lear Siegler ADM-3 的描述如下

```
adm3|lsi adm3,
    am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,
    cud1=^J, ind=^J, lines#24,
```

参数化字符串

光标寻址以及其他需要参数的字符串由参数化字符串功能使用转义符以类似于 **printf()**（请参阅 *printf*(1) ）的格式 (**%***x*) 来描述。例如，要对光标寻址，则使用两个参数给定 **cup** 功能：要寻址到的行和列（行和列从零开始编号，并参照用户可见的物理屏幕，而不是不可见的内存）。如果终端具有内存相对光标寻址，则可以通过 **mrcup** 来指示。

参数机制使用堆栈和特殊的"**%**"代码按照逆波兰式表示法 (postfix) 的方式来操作堆栈。通常，序列会将参数之

一压入堆栈，然后以某种格式打印。经常会需要执行更为复杂的操作。操作采用 postfix 格式，其操作数按照常规顺序排列。也就是说，要从第一个参数中减去 5，将使用 **%p1%{5}%-**。

“**%**”编码具有以下含义：

%% 输出“**%**”。

% [[**:**] *flags*] [*width* [*.precision*]] [**doxXs**]
与在 **printf**() 中相同，标志是 **[-+#]** 和空格。

%c 打印 **pop**() 提供 **%c**。

%p[1-9] 推动第 *i* 个参数。

%P[a-z] 将动态变量 [a-z] 设置为 **pop**()。

%g[a-z] 获取动态变量 [a-z] 并推动它。

%P[A-Z] 将静态变量 [a-z] 设置为 **pop**()。

%g[A-Z] 获取静态变量 [a-z] 并推动它。

%'*c***'** 推动 **char** 常量 *c*。

%{*nn***}** 推动十进制常量 *nn*。

%l 推动 *strlen*(*pop*())。

%+ %- %* %/ %m
算术（**%m** 是模数）：push(pop integer2 op pop integer1)，其中 integer1 表示堆栈的顶部

%& %| %^ 位运算：push(pop integer2 op pop integer1)

%= %> %< 逻辑运算：push(pop integer2 op pop integer1)

%A %O 逻辑运算：and、or

%! %~ 一元运算：*push*(op *pop*())

%i （对于 ANSI 终端）给第一个参数（如果存在一个参数）加 1，或者给前两个参数（如果存在多个参数）加 1。

%? *expr* **%t** *thenpart* **%e** *elsepart* **%;**
If-then-else； **%e** *elsepart* 是可选的； else-if 如同在 Algol 68 中一样是可能的：

%? c_1 **%t** b_1 **%e** c_2 **%t** b_2 **%e** c_3 **%t** b_3 **%e** c_4 **%t** b_4 **%e** b_5 **%;**

c_i 是条件；b_i 是主体。

如果“**-**”标志与“**%**[**doxXs**]”一起使用，则必须将一个冒号放在“**%**”和“**-**”之间，将该标志区别于二进制“**%-**”运算符。例如："**%:-16.16s**".

请考虑 Hewlett-Packard 2645，为了到达行 3 列 12，需要向其发送为 6 毫秒填充的 **\E&a12c03Y**。请注意，行和

列的顺序在此处是相反的，并且行和列用零填充为两位。因此，其 **cup** 功能是：

cup=\E&a%p2%2.2dc%p1%2.2dY$<6>

Micro-Term ACT-IV 需要前加 **^T** 的所发送当前行和列，行和列以二进制形式编码：

cup=^T%p1%c%p2%c

使用“**%c**”的设备需要能够将光标退格 (**cub1**) 并且在屏幕上将光标上移一行 (**cuu1**)。由于传输 **\n**、 **^D** 和 **\r** 并非始终是安全的（因为系统可能会将其更改或忽略），这将是必要的（处理 **terminfo** 的库函数将设置 tty 模式，使制表符永远无法扩展，因此发送 **\t** 是安全的。这对于 Ann Arbor 4080 是必不可少的）。

最后一个示例是 LSI ADM-3a，它使用通过空白字符偏移的行和列，因此：

cup=\E=%p1%'\s'%+%c%p2%'\s'%+%c

发送“**\E=**”后，它将推动第一个参数，推动空格的 ASCII 十进制值 (32)，将其相加（将所得和推到堆栈上替代两个先前的值），并将该值作为字符输出。然后为第二个参数执行相同的操作。使用堆栈时，可能会进行更为复杂的算术运算。

光标移动

如果终端能够很快地将光标复位（到屏幕的最左上角），则可以给定为 **home**；同样，快速到达左下角的方法可以给定为 **ll**；这可能涉及到随 **cuu1** 从原位向上移动，但是程序决不会自己执行该操作（除非 **ll** 执行该操作），这是因为无法假定从原位上移的效果。请注意，原位与寻址到 (0,0) 相同：屏幕而不是内存的左上角。（因此，Hewlett-Packard 终端上的 **\EH** 序列无法在不失去终端上其他一些功能的情况下用于 **home**。）

如果设备具有行或列绝对光标寻址，则可以给定为单个参数功能 **hpa**（水平绝对位置）和 **vpa**（垂直绝对位置）。有时，它们比更具一般性的双参数序列（例如 Hewlett-Packard 2645）更短，可以优先于 **cup** 使用。如果存在参数化局部移动（如“向右移动 *n* 个空格”），则可以给定为 **cud**、**cub**、**cuf** 和 **cuu**，它们用单个参数指示要移动的空格数。它们主要用于设备不具有 **cup**（如 Tektronix 4025）的情况。

如果设备需要在运行使用这些功能的程序时处于特殊模式，进入和退出该模式的代码可以给定为 **smcup** 和 **rmcup**。例如，它源自于包含多个内存页的终端（如 Concept）。如果设备仅具有内存相对光标寻址，而不具有屏幕相对光标寻址，则必须将一个屏幕大小的窗口安装到设备中，以便光标寻址正确工作。它也可用于 Tektronix 4025，其中的 **smcup** 将命令字符设置为 **terminfo** 所使用的命令字符。如果 **rmcup** 序列将不在输出 **smcup** 序列后恢复屏幕（到输出 **smcup** 之前的状态），请指定 **nrrmc**。

区域清除

如果终端可以从当前位置清除到行尾，并将光标留在原位，则应该给定为 **el**。如果终端可以从行的开头清除到当前位置（包括当前位置），并将光标留在原位，则应该给定为 **el1**。如果终端可以从当前位置清除到行尾，则应该给定为 **ed**。 **ed** 仅在行的第一列中定义（因此，如果实际的 **ed** 不可用，则可以通过请求删除大量行来对其进行模拟）。

插入和（或）删除行

如果终端可以在光标所在行之前打开一个新的空白行，则应该给定为 **il1**；这仅从行的第一个位置执行。然后，光标必须出现在新的空白行上。如果终端可以删除光标所在的行，则应给定为 **dl1**；这仅从要删除的行上的第一个位置执行。采用单个参数并插入或删除指定个行的 **il1** 和 **dl1** 的版本可以给定为 **il** 和 **dl**。

如果终端包含可设置的破坏性滚动区域（如 VT100），则可以用 **csr** 功能描述设置该区域的命令，该功能采用两个参数：滚动区域的顶部行和底部行。光标位置在使用该命令后是未定义的。使用该命令（ **sc** 和 **rc**（保存和恢复光标）命令也非常有用）可以获得插入或删除行的效果。在屏幕顶部或底部插入行的操作也可以使用 **ri** 或 **ind** 在许多终端不包含实际插入和（或）删除行功能的终端上完成，并且通常比包含这些功能的终端更快。

要确定终端是否包括破坏性滚动区域或非破坏性滚动区域，请在屏幕中间创建一个滚动区域，将数据放在滚动区域的底部行，将光标移到滚动区域的顶部行，然后执行反向索引 (**ri**) 后接删除行 (**dl1**) 或索引 (**ind**)。如果最初位于滚动区域底部行的数据由 **dl1** 或 **ind** 恢复到滚动区域中，则表明该终端包含非破坏性滚动区域。否则，它包含破坏性滚动区域。如果终端包含非破坏性滚动区域，除非 **ind**、**ri**、**indn**、 **rin**、**dl** 和 **dl1** 均模拟破坏性功能，否则不要指定 **csr**。

如果终端能够将窗口定义为所有命令都将影响到的内存部分，则应该给定为参数化字符串 **wind**。四个参数依次是内存中的起始行和结束行以及内存中的起始列和结束列。

如果终端可以保留上面的显存，则应该给定 **da** 功能；如果可以保留下面的显存，则应该给定 **db**。它们指示删除行或滚动整个屏幕可能会上移下面的非空白行，或者使用 **ri** 向后滚动可能下移非空白行。

插入和（或）删除字符

就插入和（或）删除字符操作而言，有两种基本的智能终端，它们可以使用 **terminfo.** 描述。最常用的插入和（或）删除字符操作仅影响当前行上的字符，并且将字符严格地将字符移离行尾。其他终端（如 Concept 100 和 Perkin-Elmer Owl）区分屏幕上的键入和非键入空白字符，仅在插入或删除屏幕上已删除或扩展为两个非键入空白字符的非键入空白字符时才移动。可以通过清屏并键入由光标移动分隔的文本，可以确定终端的类型。使用局部光标移动（不是空格）在 **abc** 和 **def** 之间键入“**abc def**”。然后将光标定位在 **abc** 之前并将终端置于插入模式。如果键入字符导致行的其余部分严格地移动，并使字符掉离行尾，则表明终端不区分空白字符和非键入的位置。如果 **abc** 移到 **def**，后者又一起在当前行周围移到您插入的下一行，则终端属于第二种类型，应该提供功能 **in**（表示“插入空”）。虽然它们是两个在逻辑上不相关的属性（一行与多行插入模式，以及非键入空格的特殊处理），目前还没有终端的插入模式无法用单个属性来描述。

terminfo 可以描述具有插入模式的终端和发送简单序列在当前行上打开空白位置的终端。给定为 **smir**，序列将进入插入模式。给定为 **rmir**，序列将退出插入模式。现在给定为 **ich1**，任何序列均需要在发送要插入的字符之前发送。大多数包含实际插入模式的终端将不提供 **ich1**；在此处应提供发送序列以打开屏幕位置的终端（如果终端包含这两者，则插入模式通常优先于 **ich1**。除非终端需要组合使用这两者，否则不要一起提供）。如果需要插入后填充，请在 **ip** 中将其给定为毫秒填充数（字符串选项）。任何其他需要在插入单个字符后发送的序列也可以在 **ip** 中给定。如果终端需要将这两者置于“插入模式”并且需要在每个插入的字符添加一个特殊代码，则可以给定 **smir/rmir** 和 **ich1**，它们都将使用。带有一个参数 *n* 的 **ich** 功能将插入 *n* 个空白字符。

如果填充在键入的字符之间是必需的而在插入模式下不是必需的，则在 **rmp** 中将其给定为毫秒填充数。

有时在插入模式下不必四处移动来同一行上的字符（例如，如果插入位置后有一个制表符）。如果终端允许在插

入模式下移动，则可以提供功能 **mir** 来加快这种情况下的插入速度。省略 **mir** 将只影响速度。某些终端（特别是 Datamedia）因其插入模式工作方式而不得具有 **mir**。

最后，可以指定 **dch1** 来删除单个字符，指定带一个参数 *n* 的 **dch** 来删除 *n* 个字符，以及通过给定 **smdc** 和 **rmdc** 进入和退出删除模式（终端在使用 **dch1** 时需要采用的任何模式）来删除模式。

用于清除 *n* 个字符的命令（等效于输出 *n* 个空白字符而不移动光标）可以给定为带有一个参数的 **ech**。

突出显示、下划线和可见响铃

您的设备可以具有一种或多种显示属性，用于突出显示屏幕上的选定字符。可用显示模式如下（同时列出用来设置这些显示模式的名称）：

- 闪烁的屏幕 (**blink**)
- 粗体或格外明亮的字符 (**bold**)
- 暗淡或半明亮的字符 (**dim**)
- 空白或不可见的文本 (**invis**)
- 受保护的文本 (**prot**)
- 反白显示屏幕 (**rev**)
- 备用字符集（**smacs** 用于进入该模式，**rmacs** 用于退出）。（如果在进入备用字符集模式之前必须使用命令，请在 **enacs** 或“启用备用字符集”模式中给定序列）。通过单独启用其中的任一模式可以禁用其他模式。

sgr0 应该用于禁用所有视频增强功能。由于它表示禁用某些功能的唯一方式（如 **dim** 或 **blink**），因此应该始终指定。

将一种显示方法显示为 “标准输出模式”，并使用它来突出显示错误消息和其他需要提请注意的文本。请选择一种显示形式，对比要强烈但不刺眼（建议使用反白显示加半明亮或者仅使用反白显示）。进入和退出标准输出模式的序列分别给定为 **smso** 和 **rmso**。如果进入和退出标准输出模式的代码在屏幕上留下一个或甚至两个空格（如同 TVI 912 和 Teleray 1061 的情况），则应该提供 **xmc** 来指示应该留下多少个空格。

用于开始下划线和结束下划线的序列可分别指定为 **smul** 和 **rmul**。如果设备具有给当前字符添加下划线和将光标右移一个空格的序列（如果 Micro-Term MIME），该序列可以指定为 **uc**。

包含“magic cookie”问题的终端 (**xmc**) 在收到模式设置序列时，将保存特殊“cookie”，这些序列会影响显示算法，而不是为每个字符保留额外的位。某些终端（如 Hewlett-Packard 2621）将在移到新行或者确定光标地址时自动退出标准输出模式。除非存在 **msgr** 功能，声明可以安全地在标准输出模式下移动，否则使用标准输出模式的程序应该在移动光标或发送换行符之前退出标准输出模式。

如果终端可以通过某种方式使屏幕闪烁，以无提示方式指示错误（替代响铃），则可以给定为 **flash**；它不得移动光标。有效的闪烁可以通过将屏幕更换到反白视频，填充 200 毫秒，然后将屏幕恢复到正常视频来实现。

光标不位于底部行时，如果需要比常规情况更高的可见度（例如，使非闪烁下划线变成更便于查找的块或闪烁下划线），则该序列给定为 **cvvis**。还应该指定布尔 **chts**。如果可以通过某种方式使光标完全不可见，则将其给定

为 **civis**。应该给定功能 **cnorm** ，它可取消其中任一模式的作用。

如果即使在其他情况下不叠印字符，终端仍通过使用下划线字符生成带下划线的字符（无须特殊序列），则指定功能 **ul**。对于叠印其他字符的字符将两个字符留在屏幕上的设备，请指定功能 **os**。如果叠印可以通过空白字符清除，则应该通过指定 **eo** 来指示。

如果存在设置任意模式组合的序列，则应该给定为 **sgr**（设置属性），它采用九个参数。由于相应属性的状态为打开或关闭，每个参数是 0 或非零。这九个参数依次是：standout、underline、reverse、blink、dim、bold、blank、protect、alternate character set。并非所有模式都需要 **sgr** 支持；只应支持存在其相应独立属性命令的模式。例如，假定所述的终端需要以下转义序列来启用各种模式。

tparm()

参数	属性	转义序列
	none	***\E[0m***
p1	standout	***\E[0;4;7m***
p2	underline	***\E[0;3m***
p3	reverse	***\E[0;4m***
p4	blink	***\E[0;5m***
p5	dim	***\E[0;7m***
p6	bold	***\E[0;3;4m***
p7	invis	***\E[0;8m***
p8	protect	不可用
p9	altcharset	***^O***（关）**^N**（开）

请注意，每个转义序列均需要先使用 0 来关闭其他模式，然后才能打开其自己的模式。另请注意，如上所述，*standout* 设置为 *reverse* 和 *dim* 的组合。此外，由于该终端没有 *bold* 模式，*bold* 设置为 *reverse* 和 *underline* 的组合。此外，要支持组合（如 *underline+blink*）所使用的序列将是 **\E[0;3;5m**。该终端也不具有 *protect* 模式，但这无法通过任何方式模拟，因此将忽略 **p8**。 *altcharset* 模式的不同之处在于它是**^O** 或 **^N**，这取决于它是关闭还是打开。如果要打开所有模式，序列将是：

\E[0;3;4;5;7;8m^N

现在看一看输出不同序列的情况。例如，当 **p2** 或 **p6** 为 true 时（也就是说，如果打开 *underline* 或 *bold* 模式），将输出 **;3**。如果随着其相关性写出以上序列，将提供以下内容：

序列	何时输出	**terminfo** 转换
\E[0	始终	***\E[0***
;3	如果 **p2** 或 **p6**	***%?%p2%p6%|%t;3%;***
;4	如果 **p1** 或 **p3** 或 **p6**	***%?%p1%p3%|%p6%|%t;4%;***
;5	如果 **p4**	***%?%p4%t;5%;***
;7	如果 **p1** 或 **p5**	***%?%p1%p5%|%t;7%;***
;8	如果 **p7**	***%?%p7%t;8%;***
m	始终	***m***
^N 或 **^O**	如果 **p9**，为 **^N**；否则为 **^O**	***%?%p9%t^N%e^O%;***

如果将其全部放入 **sgr** 序列，将提供：

sgr=\E[0%?%p2%p6%|%t;3%;%?%p1%p3%|%p6%
|%t;4%;%?%p5%t;5%;%?%p1%p5%
|%t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;,

请记住，必须始终指定 **sgr** 和 **sgr0**。

键盘

如果设备带有在按键时传输序列的键盘，还可以指定该信息。请注意，无法处理其键盘仅在局部工作的设备（例如，这适用于未切换的 Hewlett-Packard 2621 键）。如果可以将键盘设置为传输或不传输，请将这些序列指定为 **smkx** 和 **rmkx**。否则，将假定键盘始终会传输。

通过向左箭头、向右箭头、向上箭头、向下箭头和 Home 键发送的序列可以分别给定为 **kcub1**、**kcuf1**、**kcuu1**、**kcud1** 和 **khome**。如果存在 f0、 f1、...、f63 等功能键，它们发送的序列可以指定为 **kf0**、 **kf1**、...、**kf63**。如果前 11 个键带有缺省的 f0 至 f10，则标签可以给定为 **lf0**、 **lf1**、...、**lf10**。

可以给定其他某些特殊键传输的代码：**kll** （向下复位）、**kbs**（退格）、**ktbc**（清除所有选项卡）、**kctab** （清除该列中的制表位）、**kclr** （清屏或清除键）、 **kdch1** （删除字符）、**kdl1** （删除行）、**krmir** （退出插入模式）、**kel**（清除到行尾）、**ked** （清除到屏尾）、**kich1**（插入字符或进入插入模式）、**kil1** （插入行）、**knp** （下一页）、**kpp**（上一页）、**kind** （向前/向下滚动）、**kri**（向后/向上滚动）、**khts**（在该列中设置制表位）。此外，如果键盘具有 3 x 3 的键盘阵列（包括四个箭头键），则其他五个键可以给定为 **ka1**、 **ka3**、**kb2**、**kc1** 和 **kc3**。当需要 3 x 3 方向键盘的效果时，这些键将非常有用。其他键在上面的功能列表中定义。

功能键编程字符串可以指定为 **pfkey**、**pfloc** 和 **pfx**。屏幕标签编程字符串应指定为 **pln**。其中每个字符串采用两个参数：一个功能键标识符和一个用来编程的字符串。 **pfkey** 可使按给定键的效果用户键入给定键的效果相同；**pfloc** 可使字符串由终端在本地模式下执行；**pfx** 可使字符串传输到计算机。功能 **nlab**、**lw** 和 **lh** 定义可编程屏幕标签数及其宽度和高度。如果存在用于打开和关闭这些标签的命令，请在 **smln** 和 **rmln** 中给定。 **smln** 通常在一个或多个 **pln** 序列之后输出，以确保更改可见。

制表符和初始化

如果设备具有硬件制表符，用于前进到下一个制表位的命令可以给定为 **ht**（通常是 control-I）。左移到下一个制表位的“backtab”命令可以给定为 **cbt**。按照惯例，如果 tty 模式显示制表符正在由计算机扩展，而不是发送到

设备，由于用户可能尚未正确设置制表位，程序不应使用 **ht** 或 **cbt** （即使它们存在）。如果设备具有在设备通电时最初每隔 *n* 个空格设置的硬件制表符，则给定数字参数 **it**，显示制表符设置到的空格数。这通常由 **tput init** 用来确定是否设置硬件制表符扩展模式和是否设置制表位。如果设备具有可以在非易失性内存中保存的制表位，**terminfo** 说明可以假定它们已正确设置。如果存在用于设置和清除制表位的命令，它们可以给定为 **tbc**（清除所有制表位）和 **hts**（在每一行的当前列中设置制表位）。

其他功能包括：**is1**、**is2** 和 **is3**，设备的初始化字符串；**iprog**，要运行来初始化设备的程序的路径名；以及 **if**，包含长初始化字符串的文件的名称。这些字符串应该将设备置于与 **terminfo** 说明的其余部分一致的模式。它们必须在每次用户登录时发送到设备，并按以下顺序输出：运行程序 **iprog**；输出 **is1**；输出 **is2**；使用 **mgc**、**smgl** 和 **smgr** 设置边距；使用 **tbc** 和 **hts** 设置制表符；打印文件 **if**；最后输出 **is3**。这通常使用 **tput** 的 **init** 选项来完成。

大多数初始化均使用 **is2** 来执行。无须通过将常用序列放入 **is2** 并将特殊情况放入 **is1** 和 **is3** 来复制字符串，即可设置特殊设备模式。从完全未知状态执行重置的序列可以给定为 **rs1**、**rs2**、**rf** 和 **rs3**，它们类似于 **is1**、**is2**、**is3** 和 **if** （几种终端采用文件 **if** 和 **rf** 的模式；但是，建议的方法是使用初始化和重置字符串）。这些字符串由 **tput reset** 输出，后者在终端进入强行挤入状态时使用。只有当它们在屏幕上产生的效果并在登录时不需要时，命令通常放入 **rs1**、 **rs2**、**rs3** 和 **rf**。例如，用于将终端置于 80 列模式的命令通常属于 **is2** 的一部分，但是在某些终端上，它会在屏幕上造成恼人的故障，并且由于终端通常已经处于 80 列模式，它通常是不必要的。

如果需要使用更加复杂的序列来设置可使用 **tbc** 和 **hts** 描述的制表符，则可以将序列放入 **is2** 或 **if**。

任何边距均可以使用 **mgc** 清除（有关如何指定用于设置和清除边距的说明，请参阅下面导致移动的功能一节的边距小节）。

延迟

某些功能控制 **tty** 驱动程序中的填充。它们主要是硬拷贝终端所需要的，并且由 **tput init** 用来正确地设置 tty 模式（请参阅 *tput*(1)）。功能 **cr**、 **ind**、**cub1**、**ff** 和 **tab** 中嵌入的延迟可用于设置将在 tty 驱动程序中设置的正确延迟位。如果给定 **pb**（填充波特率），则在波特率低于 **pb** 的值时可以忽略这些值。

状态行

如果终端具有软件通常不使用的额外“状态行”，则可以指示这一事实。如果状态行被视为底部行之下的额外行，并且光标可以正常地寻址到该处（如 Heathkit H19 的第 25 行或设置为 23 行滚动区域的 VT100 的第 24 行），则应该给定功能 **hs**。转到状态行给定列并从该状态行返回的特殊字符串可以给定为 **tsl** 和 **fsl** （**fsl** 必须将光标位置保留在它位于 **tsl** 之前的相同位置。如有必要，可以将 **sc** 和 **rc** 字符串包括在 **tsl** 和 **fsl** 中来获取该效果）。功能 **tsl** 采用一个参数，即光标将移动到的状态行的列编号。

如果转义序列和其他特殊命令（如制表符）在处于状态行时工作，则可以给定标志 **eslok**。关闭状态行（或者通过其他方式清除其内容）的字符串应给定为 **dsl**。如果终端具有用于保存和恢复光标位置的命令，请将其给定为 **sc** 和 **rc**。通常假定为状态行与屏幕的其余部分（即 **cols**）具有相同的宽度。如果状态行具有不同的宽度（可能是因为终端不允许加载整个行），则可以用数字参数 **wsl** 指定以列为单位的宽度。

行图形

如果设备具有行绘制备用字符集，则将在 **acsc** 给定图形符号到字符的映射。该字符串的定义具有在 Digital VT100 终端中使用的备用字符集，并且用 AT&T 4410v1 终端中的某些字符稍作扩展。

图形符号名称	**VT100+** 字符
右指箭头	**+**
左指箭头	**,**
下指箭头	**.**
实心方块	***0***
灯笼符号	***I***
上指箭头	**-**
菱形	**‘**
棋盘（点画）	***a***
度符号	***f***
加号和（或）减号	***g***
方形板	***h***
右下角	***j***
右上角	***k***
左上角	***l***
左下角	***m***
加号	***n***
扫描线 1	***o***
横线	***q***
扫描线 9	***s***
左侧 T 型	***t***
右侧 T 型	***u***
底部 T 型	***v***
顶部 T 型	***w***
竖线	***x***
项目符号	**~**

描述新设备行图形的最佳方式是将另一列添加到上表，它包含在设备处于备用字符集模式时生成适当图形符号的新设备字符。例如：

图形符号名称	VT100+ 字符	新设备上使用的字符
左上角	***l***	***R***
左下角	***m***	***F***
右上角	***k***	***T***
右下角	***j***	***G***
横线	***q***	***,***
竖线	***x***	***.***

现在按从左到右的顺序写下字符；例如：

acsc=lRmFkTjGq\,x.

此外，通过 **terminfo**，可以定义多个字符集（请参阅下面的备用字符集一节）。

颜色操作

大多数颜色终端属于以下两类终端之一：

- **Tektronix** 式

  Tektronix 方法使用一组 N 种预定义颜色（通常是 8），从其中应用程序可以选择“当前”前景色和背景色。因此终端可以支持在屏幕上同时显示将 N 颜色混色为的 N*N 种颜色对。

- **HP** 式

  在 HP 方法中，应用程序无法独立于背景定义前景，反之亦然。应用程序必须一次定义整个颜色对。采用这种方式，可以从 2*M 种不同的颜色定义 M 种颜色对。

数字变量 **colors** 和 **pairs** 定义屏幕上可以同时显示的颜色和颜色对的数目。如果终端可以更改颜色定义（例如 Tektronix 4100 和 4200 系列的终端），则应该使用 **ccc**（可更改颜色）指定。要更改颜色定义（Tektronix 4200 方法），请使用 **initc**（将颜色初始化）。它需要四个参数：颜色编号（范围从 0 到 **colors**−1）和三个 RGB（红、绿和蓝）值或三个 HLS 颜色（色调、亮度和饱和度）值。 RGB 和 HLS 值的范围与终端相关。

Tektronix 4100 系列的终端仅使用 HLS 颜色表示法。对于这些终端（以及要以 HLS 模式运行的双模式终端），必须定义布尔变量 **hls**；它将指示 **init_color()** 函数（请参阅 *can_change_color*(3X)）先将其 RGB 参数转换为 HLS，然后再发送到终端。这样，**initc** 字符串的最后三个参数将是 HLS 值。

如果终端可以更改颜色定义，但使用不同于 RGB 和 HLS 的颜色表示法，则必须指定到 RGB 或 HLS 的映射。

如果终端支持 ANSI 转义序列设置背景和前景，它们应该分别编码为 **setab** 和 **setaf**。如果终端支持其他转义序列设置背景和前景，它们应该分别编码为 **setb** 和 **setf**。 **vidputs()** 函数（请参阅 *vidattr*(3X)）和刷新函数使用 **setab** 和 **setaf**（如果已定义）。其中每项功能均需要一个参数：颜色数。按照惯例，前八种颜色 (0–7) 按顺序映射到：黑色、红色、绿色、黄色、蓝色、洋红色、青色、白色。但是，可能发生颜色重映射，或者基础硬件可能不支持这些颜色。设备支持的任何附加颜色的映射（即映射到大于 7 的数字）由 **terminfo** 条目编写程序来控制。

要初始化颜色对（HP 方法），请使用 **initp**（初始化颜色对）。它需要七个参数：颜色对的编号（范围从 0 到 **pairs**−1）以及六个 RGB 值：三个前景值，后接三个背景值（每个包含三种颜色的组都应采用 RGB 的顺序）。

使用 **initc** 或 **initp** 时，RGB 或 HLS 参数应该分别遵循“红色、绿色、蓝色”或“色调、亮度、对比度”的顺序。要使某个颜色对成为当前颜色对，请使用 **scp**（设置颜色对）。它采用一个参数，即颜色对的编号。

某些终端（例如 PC 的大多数颜色终端模拟器）清除包含当前背景色的屏幕区域。这种情况下，应该定义 **bce**（背景色清除）。变量 **op**（初始对）包含一个序列，用于将前景色和背景色设置为它们在终端启动时所处的状态。同样，**oc** （原色）包含一个控制序列，用于将所有颜色（对于 Tektronix 方法）或颜色对（对于 HP 方法）设置为它们在终端启动时的值。

某些颜色终端用颜色替换视频属性。这些视频属性不应该与颜色组合。有关这些视频属性的信息应该装入 **ncv**（无颜色视频）变量。该变量的九个最无意义的位和视频属性之间存在一一对应关系。下表描述这种对应关系。

属性	位 位置	十进制 值	特性 设置
WA_STANDOUT	0	1	**sgr**，参数 1
WA_UNDERLINE	1	2	**sgr**，参数 2
WA_REVERSE	2	4	**sgr**，参数 3
WA_BLINK	3	8	**sgr**，参数 4
WA_DIM	4	16	**sgr**，参数 5
WA_BOLD	5	32	**sgr**，参数 6
WA_INVIS	6	64	**sgr**，参数 7
WA_PROTECT	7	128	**sgr**，参数 8
WA_ALTCHARSET	8	256	**sgr**，参数 9
WA_HORIZONTAL	9	512	**sgr1**，参数 1
WA_LEFT	10	1024	**sgr1**，参数 2
WA_LOW	11	2048	**sgr1**，参数 3
WA_RIGHT	12	4096	**sgr1**，参数 4
WA_TOP	13	8192	**sgr1**，参数 5
WA_VERTICAL	14	16384	**sgr1**，参数 6

当特定视频属性不应与颜色一起使用时，请将相应的 **ncv** 位设置为 1；否则将其设置为 0。要确定将装入 **ncv** 变量的信息，请添加与无法与颜色共存的属性相对应的十进制值。例如，如果终端使用颜色来模拟反白显示（位编号 2 和十进制值 4）和粗体（位编号 5 和十进制值 32），所得的 **ncv** 值将是 36 (4 + 32)。

其他信息

如果终端需要空（零）字符之外的字符作为填充，它可以给定为 **pad**。仅使用 **pad** 字符串的第一个字符。如果终端不包含填充字符，请指定 **npc**。

如果终端可以向上或向下移动半行，则可以使用 **hu**（上移半行）和 **hd**（下移半行）来指定。它主要用于硬拷贝设备上的上标和下标。如果硬拷贝设备可以弹出到下一页（换页符），则将其给定为 **ff** （通常是 control-L）。

如果存在将给定字符重复给定次数的命令（以节省传输大量相同字符所需的时间），则可以使用参数化字符串 **rep** 来指示。第一个参数是要重复的字符，第二个参数是要重复的次数。因此，**tparm(repeat_char, ’x’, 10)** 与 **xxxxxxxxxx** 相同。

如果终端包含可设置的命令字符（如 Tektronix 4025），则可以使用 **cmdch** 指示。将选择在所有功能中均使用的原型命令字符。该字符在 **cmdch** 功能中给定来进行标识。某些系统上支持以下惯例：如果存在环境变量 **CC**，原型字符的所有实例均将替换为 **CC** 中的字符。

不表示特定已知终端类型的终端说明（如 *switch*、*dialup*、*patch* 和 *network*）应该包括 **gn**（一般）功能，以便程序发出警告，指示它们不知道如何与终端通信（该功能不适用于知道其转义序列的虚拟终端说明）。如果终端是虚拟终端协议支持的终端之一，则终端编号可以给定为 **vt**。要在执行读取之前传输的 line-turn-around 序列应该在 **rfi** 中指定。

如果设备使用 XON/XOFF 握手进行流控制，请给定 **xon**。仍应该包括填充信息，以便函数可以做出更佳的成本决策，但是将不传输实际的填充字符。用于打开和关闭 XON/XOFF 握手的序列可以在 **smxon** 和 **rmxon** 中给定。如果用于握手的字符不是 **^S** 和 **^Q**，它们可以使用 **xonc** 和 **xoffc** 指定。

如果终端包含充当 Shift 键的"转换键"，并设置传输的任何字符的第 8 位，则可以使用 **km** 指示该事实。否则，软件将假定第 8 位是奇偶校验位，并且通常会将其清除。如果存在用于打开和关闭该"元模式"的字符串，它们可以给定为 **smm** 和 **rmm**。

如果终端包含屏幕上无法同时容纳的多行内存，则可以使用 **lm** 指示内存行数。 **lm#0** 值指示行数不是固定的，但是仍有其他内存无法在屏幕上显示。

控制连接到终端的辅助打印机的介质复制字符串可以给定为：

mc0 打印屏幕的内容。
mc4 关闭打印机。
mc5 打开打印机。

打印机打开时，发送到终端的所有文本均将发送到打印机。变量 **mc5p** 采用一个参数，并在字符数与该参数值相同的情况下保持打印机的打开状态，然后将其关闭。该参数不应超过 255。如果打印机打开时文本没有显示在终端屏幕上，请指定 **mc5i**（无提示打印机）。只要 **mc5p** 有效，所有文本（包括 **mc4**）均将以透明的方式传递到打印机。

特殊情况

terminfo 使用的工作模式很好地适合大多数终端。但是，某些终端并不完全匹配该模式，而需要 **terminfo** 的特殊支持。它们并不是终端中的缺陷，而仅仅是工作模式和实际硬件之间的不同。它们可以是特殊设备，或者出于某种原因，未实现 **terminfo** 模型的所有功能。

无法显示波浪符 (~) 的终端（如某些 Hazeltine 终端）应该指示 **hz**。

忽略紧接 **am** 换行后的换行符的终端（如 Concept 100）应该指示 **xenl**。在收到其他字符之前留在最右侧列（而不是在收到最右侧字符立即换行）的终端（如 VT100）也应该指示 **xenl**。

如果使用 **el** 来去除标准输出（而不是在其上写入常规文本），则应该给定 **xhp**。

其制表符将所有字符移到空白字符处的 Teleray 终端应该指示 **xt**（破坏性制表符）。该功能也表示无法将光标定位在"magic cookie"之上。因此，要清除标准输出模式，有必要使用删除和插入行。

对于不传输转义符或 control-C 字符的 Beehive Superbee 终端，请指定 **xsb**，指示 f1 键用于转义符，f2 键用于 control-C。

类似终端

如果存在两个类似终端，一个终端可以定义为与另一个终端类似，但另一个终端具有某些例外。字符串功能 **use** 可以用类似终端的名称来给定。在 **use** 之前给定的功能将覆盖终端类型中由 **use** 调用的功能。通过将功能名@ 放在字符串 **use** 之前，可以取消该功能。例如，条目：

```
att4424-2|Teletype 4424 in display function group ii,
    rev@, sgr@, smul@, use=att4424,
```

定义不具有 **rev**、**sgr** 和 **smul** 功能，因此无法执行突出显示的 AT&T 04424 终端。这对于终端的不同模式以及不同的用户首选项非常有用。可以给定多个 **use** 功能。

打印机功能

通过 **terminfo** 数据库，可以定义打印机和终端的功能。可用于打印机的功能包括在上面定义的功能一节的列表中。

舍入值

由于参数化字符串功能仅使用整数值， **terminfo** 设计人员应该创建预期已舍入的数字值的字符串。应用程序设计人员应该注意到这点，并且应该始终在将值用于参数化的字符串功能之前将其舍入到最接近的整数。

打印机分辨率

打印机的分辨率定义为它可以达到的最小字符间距。通常，水平分辨率和垂直分辨率不相关。因此，打印机的垂直分辨率可以通过测量在连续打印基线之间可达到的最小距离来确定，而水平分辨率则可以通过测量在连续打印的相同字符的左边缘之间可达到的最小距离来确定。

假定所有打印机均能够以均匀的水平分辨率和垂直分辨率进行打印。 **terminfo** 当前提供的打印视图是均匀矩阵中的打印视图。所有字符均在相对于矩阵中每个“单元格”的固定位置打印；此外，每个单元格具有通过分辨率所示的最小水平和垂直步进大小而给定的相同大小（如下文所述，单元格大小可以更改）。

许多打印机具有“比例打印”功能，即水平间距取决于打印的最后一个字符的大小。虽然 **terminfo** 提供了足够的功能定义来支持应用程序模拟比例打印，但它没有利用该功能。

打印机不但必须打印足够靠近的字符（如水平和垂直分辨率所示），还必须能够从先前位置移到距离最小距离整数倍的位置。这样，打印的字符就可以分开一定的距离，该距离是最小距离的整数倍，最大值是单个页面的长度或宽度。

某些打印机可以根据不同的“模式”使用不同的分辨率。. 在“常规模式”下，假定现有 **terminfo** 功能按照列和行来工作，就像视频终端一样。因此，旧的 **lines** 功能将以行为单位提供页面的长度，而 **cols** 功能将以列为单位提供页面的宽度。在“微观模式”下，许多 **terminfo** 功能按照行和列的递增量来工作。对于某些打印机，微观模式可以与常规模式并存，因此所有功能都可以同时工作。

指定打印机分辨率

可以通过多种方式指定打印机的打印分辨率。每种方式均将分辨率指定为每个距离的最小步进数：

典型的最小步进数	
orhi	水平方向每英寸步进数
orvi	垂直方向每英寸步进数
orc	每列步进数
orl	每行步进数

以常规模式打印时，除下面所述的特殊情况外，所打印的每个字符将导致移至下一列；所移动的距离与每列分辨率相同。某些打印机将在最右侧位置打印字符时导致自动移至下一行；垂直移动的距离与每行分辨率相同。在微观模式下打印时，这些距离可以不同，并且对于某些打印机可能是零。

打印后的自动移动	
常规模式：	
orc	水平移动的步进数
orl	垂直移动的步进数
微观模式：	
mcs	水平移动的步进数
mls	垂直移动的步进数

某些打印机能够打印宽字符。在常规模式下打印宽字符时所移动的距离可能与打印常规宽度字符时不同。在微观模式下打印宽字符时所移动的距离也可能与在微观模式下打印常规字符时不同，但假定差异是相关的：如果常规字符在常规模式或微观模式下移动相同的距离 (**mcs=orc**)，则宽字符在常规模式或微观模式下也移动相同的距离。这不意味着常规字符距离一定与宽字符距离相同，而只表示距离不随着从常规模式到微观模式的切换而更改。但是，如果常规字符在微观模式下移动的距离不同于在常规模式下移动的距离 (**mcs<orc**)，则假定微观模式距离对于微观模式下打印的宽字符是相同的，如下表所示。

打印宽字符后的自动移动	
常规模式或微观模式 (**mcs = orc**)：	
widcs	水平移动的步进数
微观模式 (**mcs < orc**)：	
mcs	水平移动的步进数

可能存在一些控制序列，可更改每英寸的列数（字符间距）以及每英寸的行数（行间距）。如果使用这些序列，打印机的分辨率将更改，但是更改的类型取决于打印机：

更改字符/行间距	
cpi	更改字符间距
cpix	如果设置，**cpi** 将更改 **orhi**；否则，将更改 **orc**
lpi	更改行间距
lpix	如果设置，**lpi** 将更改 **orvi**；否则，将更改 **orl**
chr	更改每列步进数
cvr	更改每行步进数

cpi 和 **lpi** 字符串功能各自采用单个参数，它们分别是列（或字符）间距和每英寸行数。 **chr** 和 **cvr** 字符串功能各自采用单个参数，它们分别是每列步进数和每行步进数。

如果使用这些字符串中的任何控制序列，则暗示着 **orc**、**orhi**、**orl** 和 **orvi** 中的某些值将出现更改。此外，打印宽字符时移动的距离 **widcs** 将相对于 **orc** 变化。在微观模式下打印字符时所移动的距离 **mcs** 以类似的方式进行变化，一个例外是：如果距离是 0 或 1，则假定无更改。

使用 **cpi**、**lpi**、**chr** 或 **cvr** 的程序应该重新计算打印机分辨率（并且应该重新计算其他值）。请参阅下面的更改打印分辨率的效果一节。

更改字符/行间距的效果	
之前	之后
使用 **cpi** 和 **cpix** 清除:	
orhi'	***orhi***
orc'	***orc = orhi /*** *Vcpi*
使用 **cpi** 和 **cpix** 设置:	
orhi'	***orhi = orc **** *Vcpi*
orc'	***orc***
使用 **lpi** 和 **lpix** 清除:	
orvi'	***orvi***
orl'	***orl = orvi /*** *Vlpi*
使用 **lpi** 和 **lpix** 设置:	
orvi'	***orvi = orl **** *Vlp*
orl'	***orl***
使用 **chr:**	
orhi'	***orhi***
orc'	*Vchr*
使用 **cvr:**	
orvi'	***orvi***
orl'	*Vcvr*
使用 **cpi** 或 **chr:**	
widcs'	***widcs = widcs' * orc / orc'***
mcs'	***mcs = mcs' * orc / orc'***

Vchr、 *Vcpi*、 *Vcvr* 和 *Vlpi* 分别用于 **chr**、 **cpi**、 **cvr** 和 **lpi** 的参数。撇号 (**'**) 指示旧值。

引发移动的功能

在下面的说明中，“移动”是指“当前位置”的移动。对于视频终端，它可能是光标；对于某些打印机，它可能回车位置。其他计算机具有不同的等效对象。通常，当前位置是字符在打印时的显示位置。

terminfo 具有引发移动整列数或整行数的控制序列的字符串功能。它还具有引发移动最小步进数的控制序列的字符串功能。

用于移动的字符串功能	
mcub1	左移 1 步
mcuf1	右移 1 步
mcuu1	上移 1 步
mcud1	下移 1 步
mcub	左移 *N* 步
mcuf	右移 *N* 步
mcuu	上移 *N* 步
mcud	下移 *N* 步
mhpa	从左侧移动 *N* 步
mvpa	从顶端移动 *N* 步

最后六个字符串各自采用单个参数 *N* 。

有时，移动范围限于小于页面的宽度或长度。此外，某些打印机不接受向当前位置左侧的绝对移动。 **terminfo** 具有指定这些限制的功能。

移动限制	
mjump	限制使用 **mcub1**、 **mcuf1**、 **mcuu1**、 **mcud1**
maddr	限制使用 **mhpa**、 **mvpa**
xhpa	如果设置，**hpa** 和 **mhpa** 将无法左移
xvpa	如果设置，**vpa** 和 **mvpa** 将无法上移

如果需要打印机处于“微观模式”，上述的移动功能才能正常工作，则有一些定义的字符串包含用于进入和退出该模式的控制序列。对于回车导致自动恢复到常规模式的打印机，它们可以使用布尔值。

进入和（或）退出微观模式	
smicm	进入微观模式
rmicm	退出微观模式
crxm	使用 **cr** 退出微观模式

对于不同的计算机，在最右侧打印字符时的移动将有所不同。某些打印机不移动，有些打印机移到下一行的开头，其他打印机则移到同一行的开头。 **terminfo** 具有用于描述所有三种情况的布尔功能。

字符在最右侧位置打印后的移动	
sam	自动移到同一行的开头

可以将某些打印机置于与正常移动方向相反的模式。当没有向左或向上移动功能时，该模式特别有用，因为这些功能可以通过反向移动功能和向右或向下移动功能来构建。最好让应用程序来构建向左或向上功能，而不要在 **terminfo** 数据库中输入这些功能。这样，几个反向移动就可以组合在一起，而不会干预那些退出和重新进入反向模式的步骤，这些步骤是多余的。

进入和（或）退出反向模式	
slm	水平反向移动
rlm	水平恢复移动
sum	垂直反向移动
rum	垂直恢复移动
当进行水平反向移动时：	
mcub1	右移 1 步
mcuf1	左移 1 步
mcub	右移 *N* 步
mcuf	左移 *N* 步
cub1	右移 1 列
cuf1	左移 1 列
cub	右移 *N* 列
cuf	左移 *N* 列
当进行垂直反向移动时：	
mcuu1	下移 1 步
mcud1	上移 1 步
mcuu	下移 *N* 步
mcud	上移 *N* 步
cuu1	下移 1 行
cud1	上移 1 行
cuu	下移 *N* 行
cud	上移 *N* 行

反向移动模式不会影响 **mvpa** 和 **mhpa** 绝对移动功能。但是，反向垂直移动模式还应该反转当字符在最右侧位置打印时进行的“换行”操作。因此，已定义标准 **terminfo** 功能 **am** 的打印机应该在字符以反向垂直移动模式在最右侧位置打印时经历向上一行开头的移动。

以反向移动模式使用其他任何功能时的移动尚未定义；因此，在使用其他移动功能之前，程序必须退出反向移动模式。

移动功能列表还包括最后两项其他功能。其中一个是在使用某些控制符（如换行符或换页符）时将当前位置移到行开头的打印机所需要的。另一个用于挂起通常在打印字符后进行的移动的功能。

其他移动字符串	
docr	导致 **cr** 的控制符列表
zerom	防止在打印下一个字符后的自动移动

边距

terminfo 提供了两个用于在终端上设置边距的字符串：一个用于左边距，一个用于右边距。但是，打印机还有另外两个边距：每一页的上边距和下边距。此外，某些打印机要求不使用移动字符串来将当前位置移到边距处并将边距固定在此处，而要求指定与当前位置无关的边距距离。因此， **terminfo** 提供了六个附加字符串来定义打印机的边距。

设置边距	
smgl	在当前列处设置左边距
smgr	在当前列处设置右边距
smgb	在当前行处设置下边距
smgt	在当前行处设置下边距
smgbp	在行 *N* 处设置下边距
smglp	在列 *N* 处设置左边距
smgrp	在列 *N* 处设置右边距
smgtp	在行 *N* 处设置上边距

最后四个字符串采用一个或多个字符串，它们提供要设置的边距的位置。如果 **smglp** 和 **smgrp** 均已设置，则各自采用单个参数 *N*，它分别提供左边距和右边距的列数。如果 **smgtp** 和 **smgbp** 均已设置，则分别用于设置上边距和下边距：**smgtp** 采用单个参数 *N*，即上边距的行数；但是 **smgbp** 采用两个参数 *N* 和 *M*，它们提供下边距的的行数，第一个参数从页顶端计算，第二个参数从底端计算。它支持不同制造商的计算机中两者不同指定下边距的方法。为具有可设置下边距的打印机编码 **terminfo** 条目时，根据具体的打印机，只应使用第一个或第二个参数。编写使用 **smgbp** 来设置下边距的应用程序时，必须给定这两个参数。

如果仅设置 **smglp** 和 **smgrp** 中的一个，它将采用两个参数，依次是左边距和右边距的列数。同样，如果仅设置 **smgtp** 和 **smgbp** 中的一个，它将采用两个参数，它们依次提供从页面顶端计算的上边距和下边距。因此，当为需要同时设置左右边距或上下边距的打印机编码 **terminfo** 条目时，只应定义 **smglp** 和 **smgrp** 或 **smgtp** 和 **smgbp** 中的一个；另一个应该留为空白。编写将使用这些字符串功能的应用程序时，应该首先检查参数对，确定是设置了参数对中的每一个还是仅设置了一个，然后应该相应地使用它们。

在计算行数或列数，第零行是顶行，第零列是最左侧的列。如果 **smgbp** 的第二个参数的值为零，则指示该页的底端行。

所有边距均可以使用 **mgc** 清除。

阴影、斜体、宽字符、上标、下标

五组字符串描述打印机的增强文本打印功能。

增强打印	
sshm	进入阴影打印模式
rshm	退出阴影打印模式
sitm	进入斜体模式
ritm	退出斜体模式
swidm	进入宽字符模式
rwidm	退出宽字符模式
ssupm	进入上标模式
rsupm	退出上标模式
supcs	可用作上标的字符列表
ssubm	进入下标模式
rsubm	退出下标模式
subcs	可用作下标的字符列表

如果打印机在要阴影打印的每个字符之前需要 **sshm** 控制序列，则 **rshm** 字符串将留为空白。因此，在 **sshm** 中找到控制序列，但在 **rshm** 中没有找到控制序列的程序应该在每个要阴影打印的字符前使用 **sshm** 控制序列；否则，应该在要阴影打印的字符集前使用一次 **sshm**，后接 **rshm**。对于每个 **sitm-ritm**、 **swidm-rwidm**、 **ssupm-rsupm** 和 **ssubm-rsubm** 对也是相同的情况。

terminfo 还具有用于打印粗体文本的功能 (**bold**)。阴影打印和粗体打印在将文本“暗化”方面比较相似，许多打印机会通过稍微不同的方式产生这两种打印效果。通常，粗体打印通过将相同字符叠印一次或多次来实现。阴影打印通常同样采用叠印，但是会略微向上和（或）向侧面移动，使字符显得“更丰满”。

假定增强打印模式是独立的模式，因此可以阴影打印斜体的上标。

如上所述，打印宽字符后的自动移动量应该在 **widcs** 中给定。

如果只有一部分可打印的 ASCII 字符可以打印为上标或下标，它们应该分别在 **supcs** 或 **subcs** 字符串中列出。如果 **ssupm** 或 **ssubm** 字符串包含控制序列，但是相应的 **supcs** 或 **subcs** 字符串为空，则假定所有可打印的 ASCII 字符可用作上标或下标。

假定打印上标或下标之后的自动移动与标准字符相同。因此，打印以下三个示例中的任何一个将导致等效移动：

Bi $\mathbf{B_i}$ $\mathbf{B^i}$

请注意，现有 **msgr** 布尔功能描述在“标准输出模式”是否可以使用移动控制序列。该功能已扩展，包括此处添加的增强打印模式。应该为接受任何移动控制序列，而不影响阴影、斜体、加宽、上标或下标打印的打印机设置

msgr。

相反，如果未设置 **msgr**，程序应该在尝试任何移动之前结束这些模式。

备用字符集

除了用于定义行图形（在上面插入和（或）删除字符一节的行图形中说明）之外， **terminfo** 还可用于定义备用字符集。以下功能涉及到包含多个可选择或可定义字符集的打印机和终端：

备用字符集	
scs	选择字符集 *N*
scsd	开始字符集 *N*（*M* 个字符）的定义
defc	定义字符 *A*，宽度 *B* 点，下降字符 *D*
rcsd	结束字符集 *N* 的定义
csnm	字符集名称列表
daisy	打印机已手动更换打印轮

scs、**rcsd** 和 **csnm** 字符串采用单个参数 *N* ，它是标识字符集且介于 0 到 63 的数字。 **scsd** 字符串页采用参数 *N* 和另一个参数 *M* ，后者提供字符集中的字符数。 **defc** 字符串采用三个参数：*A* 提供字符的 ASCII 代码表示形式， *B* 提供以点为单位的字符宽度，*D* 是零或一（取决于字符是否是“降序符”）。 **defc** 字符串还后接“图像数据”字节的字符串，它描述字符的外观（请参阅下文）。

字符集 0 是在打印机初始化后存在的缺省字符集。并非所有打印机都有 64 个字符集；使用 **scs** 不选择可用字符集的参数，会导致 **tparm()** 返回空指针（请参阅 *tigetflag*(3X)）。

如果字符集需要在使用之前先定义，则将在定义字符集之前使用 **scsd** 控制序列，在定义之后使用 **rcsd**。与不适用的 *N* 一起使用时，它们也会导致 **tparm()** 返回空指针。如果字符集在定义后仍需要选择， **scs** 控制序列应该在 **rcsd** 控制序列之后出现。通过以 **tparm()** 调用中的字符集编号检查 **scs**、**scsd** 和 **rcsd** 字符串中的每一个字符串，程序可以确定需要这三个字符串中的哪一个。

在使用 **scsd** 和 **rcsd** 字符串之间，应该使用 **defc** 字符串来定义每个字符。要在 **terminfo** 所述的打印机上打印任何字符，ASCII 代码将发送到该打印机。对于备用字符集中的字符和“常规”字符均是如此。因此，字符定义包括表示该字符的 ASCII 代码。此外，将给定以点为单位的字符宽度，并指示字符是否应该在打印行下下降（如大多数字符集中的小写字符“**g**”）。以点为单位的字符宽度还指示 **defc** 字符串之后的图像数据字节数。这些图像数据字节指示应该在点阵图案中的何处施墨“绘制”该字符；这些字节的数目及其形式在下面的点阵图形一节中定义。

terminfo 条目的创建者按编号引用各个字符集是最容易的方式；但是对于应用程序开发人员，这些编号将毫无意义。 **csnm** 字符串通过为每个编号提供名称来缓解了这一问题。

与 **tparm()** 调用中的字符集编号一起使用时， **csnm** 字符串将生成等效的名称。这些名称只应用作参考。虽然为打印机创建 **terminfo** 条目的任何人员都应使用与打印机用户文档中的名称一致的名称，但并不意味着任何命名惯例。应用程序开发人员应该允许用户按编号（让用户检查 **csnm** 字符串来确定正确的编号）或按名称（应用程序检查 **csnm** 字符串以确定相应的字符集编号）指定字符集。

这些功能可能仅用于点阵打印机。如果它们不可用，则不应该指定这些字符串。对于手动更换打印轮或字体盒的打印机，将设置布尔 **daisy** 。

点阵图形

点阵打印机通常具有重现光栅图形的功能。通过三个数字功能和三个字符串功能，程序将绘制光栅图形图像，这独立于点阵打印机的类型以及打印机一次可处理的针数或点数。

点阵图形	
npins	打印头中的针数 *N*
spinv	针垂直间距（每英寸的针数）
spinh	点水平间距（每英寸的点数）
porder	将软件位与打印头的针匹配
sbim	开始打印位图图形，宽度 *B* 位
rbim	停止打印位图图形

sbim 字符串采用单个参数 *B* ，即以点为单位的图像宽度。

terminfo 提供的点阵或光栅图形的模型与大多数点阵打印机所使用的技术类似：假定打印机打印头的每一次通过都将生成一个 *N* 点高 *B* 点宽的点阵。它通常是由点组成的、宽而短的矩形。根据具体的打印机，该矩阵的高度（以点为单位）将有所不同；它在 **npins** 数字功能中给定。矩形的大小（一英寸的分数）也将有所不同；这可以从 **spinv** 和 **spinh** 数字功能推导。利用这三个值，任何应用程序都可以将完整的光栅图形图像分为几个水平的条带，并且可能为表示不同的水平和垂直点间距而互插。

sbim 和 **rbim** 字符串分别开始和结束点阵图像。 **sbim** 字符串采用单个参数，它提供以点为单位的点阵宽度。一个“图像数据字节”序列将在 **sbim** 字符串之后、 **rbim** 字符串之前发送到打印机。字节数是点阵宽度的整数倍；该倍数和每个字节的形式在下面所述的 **porder** 字符串中确定。

porder 字符串是针编号的逗点分隔列表（可能后接数字偏移量）。偏移量如果给定，将用分号与列表隔开。每个针编号在列表中的位置对应于 8 位数据类型中的一个数位。各个针从 1 到 **npins** 连续编号，1 表示顶端针。请注意，“针”一词在本文使用并不严格；“喷墨”点阵打印机没有针，但仍可视为具有等效的方法来将一个墨点施加到纸上。 **porder** 中位的位置 8 个一组，每组中的第一个位置是最重要的位，处于最后一个位置的位不是很重要。应用程序按照组在 **porder** 中的顺序生成 8 位字节。

应用程序通过内部图像计算“图像数据字节”：将每次打印头通过中的垂直点位置映射到 8 位字节，使用 1 位表示应该施墨的位置，使用 0 表示不应施墨的位置。这可以通过指定负的针编号来反转（0 位表示有墨，1 位表示无墨）。如果在 **porder** 中跳过某一位置，则使用 0 位。如果某一位置具有小写的“**x**”而不是针编号，则在跳过的位置使用 1 位。为了确保一致性，小写的“**o**”可用于表示 0 填充、跳过位。必须在 **porder** 中使用或跳过 8 位位置的倍数；否则，最后一个字节的低序位将设置为 0。偏移量如果给定，将添加到每个数据字节；偏移量可以为负。

一些示例可能有助于阐明 **porder** 字符串的用法。 AT&T 470、AT&T 475 和 C.Itoh 8510 打印机为图形提供八针。这些针从顶部到底部在一个字节中按从最不重要到最重要的顺序以 8 位进行标识。这些打印机的 **porder** 字符串

将是 **8,7,6,5,4,3,2,1**。 AT&T 478 和 AT&T 479 打印机也为图形提供八针。但是，这些针以相反的顺序标识。这些打印机的 **porder** 字符串将是 **1,2,3,4,5,6,7,8**。

AT&T 5310、 AT&T 5320、Digital LA100 和 Digital LN03 打印机为图形提供六针。这些针从顶部到底部用十进制值 1、 2、4、8、16 和 32 来标识。虽然这些十进制值将通过值 63 进一步偏移，但它们对应于 8 位字节的低六位。这些打印机的 **porder** 字符串将是 **,,6,5,4,3,2,1;63**，它等效于 **o,o,6,5,4,3,2,1;63**。

更改打印分辨率的效果

如果使用了更改字符间距或行间距的控制序列，则针间距或点间距可能会更改：

更改字符/行间距	
cpi	更改字符间距
cpix	如果设置，**cpi** 将更改 **spinh**
lpi	更改行间距
lpix	如果设置，**lpi** 将更改 **spinv**

使用 **cpi** 或 **lpi** 的程序应该重新计算点间距：

更改字符/行间距的效果

之前	之后
使用 **cpi** 和 **cpix** 清除：	
spinh'	***spinh***
使用 **cpi** 和 **cpix** 设置：	
spinh'	***spinh = spinh' * orhi / orhi'***
使用 **lpi** 和 **lpix** 清除：	
spinv'	***spinv***
使用 **lpi** 和 **lpix** 设置：	
spinv'	***spinv = spinv' * orhi / orhi'***
使用 **chr**：	
spinh'	***spinh***
使用 **cvr**：	
spinv'	***spinv***

orhi' 和 **orhi** 分别是在使用 **cpi** 之前和使用 **cpi** 之后的水平分辨率值（以每英寸步数为单位）。同样，**orvi'** 和 **orvi** 分别是在使用 **lpi** 之前和使用 **lpi** 之后的垂直分辨率值（以每英寸步数为单位）。因此，点阵图形的更改（以每英寸步数为单位）跟随打印机分辨率的更改（以每英寸步数为单位）。

打印质量

许多点阵打印机均可以更改打印文本的点间距，以产生 仿信函质量或 草稿质量打印效果。由于打印速度通常会随着质量的提高而降低，因此选择某一质量的功能是非常重要的。三个字符串描述这些功能：

打印质量	
snlq	设置仿信函品质打印
snrmq	设置常规品质打印
sdrfq	设置草稿品质打印

功能按照质量递减的顺序列出。如果打印机不具有所有三个质量级别，则应该将相应的字符串留为空白。

打印速率和缓冲区大小

由于没有标准协议可用于使程序与打印机同步，并且现代的打印机可以在打印之前缓冲数据，因此程序无法确定已打印的内容。通过两个数字功能，程序可以估计已打印的内容。

打印速率和（或）缓冲区大小	
cps	常规打印速率（每秒的字符数）
bufsz	缓冲区容量（字符数）

cps 是打印机打印字符的额定或平均速率；如果未给定该值，则应该估计速率是主导波特率的十分之一。 **bufsz** 是在确保打印先前字符之前缓冲的最大后继字符数（假定使用了正确的流控制）。如果未给定该值，则假定打印机不缓冲字符，但会在收到这些字符时将其打印。

例如，如果打印机具有 1000 个字符的缓冲区，则发送字母“**a**”后接 1000 个附加字符一定会打印字母“**a**”。如果同一台打印机以每秒 100 个字符的速率打印，它会使用 10 秒时间打印缓冲区中的所有字符，如果缓冲区未满，所用时间将更少。通过跟踪发送到打印机的字符并且知道打印速率和缓冲区大小，程序可以将其自身与打印机同步。

请注意，大多数打印机制造商在广告中使用的是最大打印速率，而不是额定打印速率。要获取输入 **cps** 的值，较好的方法是生成几页文本，计算可打印字符数，然后确定打印文本所用的时间。

使用这些值的应用程序应该了解打印速率的变动性。无嵌入控制序列的短行纯文本的打印速率可能会以接近广告打印速率，并且可能比 **cps** 中的速率更快。包含大量控制序列的图形数据或者非常长的文本的打印速率将大大低于广告速率，并低于 **cps** 中的速率。如果应用程序使用 **cps** 来确定打印机应使用多少时间来打印一个文本块，则应用程序应该增大估计值。如果应用程序使用 **cps** 来确定已经打印多少文本，则应该减少估计值。因此，为了让希望看到所有输出均位于其正确位置的用户满意，应用程序将会输出错误结果。

选择终端

如果定义了环境变量 **TERMINFO**，则使用 Curses 的任何程序都将先检查本地终端定义，然后再检查标准的位置。例如，如果 **TERM** 设置为 **att4424**，缺省情况下将在以下路径查找编译的终端定义

a/att4424

（它位于实现专用的目录中）。

（“**a**”从 **att4424** 的第一个字母复制，用于避免创建巨大的目录）。但是，如果 **TERMINFO** 设置为 **$HOME/myterms**， Curses 将首先检查

$HOME/myterms/a/att4424

如果失败，则检查缺省路径名。

这对于开发试验性定义或者在实现定义缺省数据库中的写权限不可用时非常有用。

如果已设置 **LINES** 和 **COLUMNS** 环境变量，或者程序正在窗口环境中执行，环境中的行和列信息将覆盖 **terminfo** 读取的信息。

应用程序用法

要编写终端说明，最有效的方法是模仿 **terminfo** 中对类似终端的说明，并逐步建立说明（使用通过面向屏幕的编辑器创建的部分说明），最后检查其正确性。要轻松地测试新终端说明，可以将环境变量 **TERMINFO** 设置为包含编译说明的目录的路径名，程序将在此处查找，而不是在 **terminfo** 数据库中查找。

设备别名惯例

必须给每个设备分配一个名称，如 **vt100**。应该按照以下惯例选择设备名称（除完整名称外）。由于连字符保留在添加指示特殊模式的后缀时使用，因此名称不应包含连字符。

这些特殊模式可以是硬件可以进入或用户首选的模式。要给特定设备分配特殊模式，请将由连字符和模式指示符组成的后缀追加到设备名称。例如，**-w** 后缀表示 宽模式；如果指定，允许 132 列（而不是标准的 80 列）的宽度。因此，如果要使用设置为宽模式的 VT100 设备，请将该设备命名为 **vt100-w**。在可能情况下使用以下后缀：

后缀	含义	示例
-w	宽模式（多于 80 列）	***5410-w***
-am	带自动边距（通常是缺省值）	***vt100-am***
-nam	无自动边距	***vt100-nam***
-*n*	屏幕上的行数	***2300-40***
-na	无箭头键（留在本地）	***c100-na***
-*n****p***	内存页数	***c100-4p***
-rv	反白显示	***4415-rv***

终端定义的变化

terminfo 中条目的创建方式是由实现定义的。

可以通过多种方法来编写 **terminfo** 条目。最低程度的条目可能允许应用程序使用 Curses 来操作终端。如果条目已增强，描述终端的多项功能，则应用程序可以使用 Curses 来调用这些功能，并且可以利用 Curses 中的优化，进而更加高效地操作。对于大多数终端，已经编写了最优的 **terminfo** 条目。

外部语言环境影响

环境变量

CC 指定原型命令字符的替换字符。请参阅插入和（或）删除行一节的其他信息小节中的 **cmdch**。

COLUMNS
指定可以覆盖 **terminfo** 中列信息的列信息。请参阅选择终端一节。

LINES 指定可以覆盖 **terminfo** 中行信息的行信息。请参阅选择终端一节。

TERM 指定当前终端的名称。请参阅选择终端一节。

TERMINFO
指定本地终端定义的备用位置。如果未在 **$TERMINFO/?/*** 中找到 **TERM** 或者未设置 **TERMINFO**，则将在缺省位置 **/usr/lib/terminfo/?/*** 查找值。请参阅选择终端一节。

另请参阅

tic(1)、 untic(1)、 tgetent(3X)、 tigetflag(3X)、 term(4)、 term(5)。

ANSI 标准 X3.64-1979。

X/Open 系统接口定义第 4 期第 2 版。

名称

ttys - 终端控制数据库文件，用于信任系统

概要

/tcb/files/ttys

说明

系统支持单个终端控制数据库，该数据库包含每个可以登录到系统本地终端的条目。身份验证程序使用终端控制数据库中的信息来决定是否允许来自该终端的登录。出于信息考虑，而保留附加字段。

终端控制数据库文件的格式与其他系统身份验证数据库文件相同。关于文件格式的详细信息，请参阅 *authcap* (4)。该文件由关键字字段标识符及那些字段的值组成。支持的关键字标识符及其使用情况包括：

t_devname 本字段为该条目定义了终端设备名。终端设备应该包含在 **/dev** 目录中，因此无须指定该前缀。如果终端条目描述了 **/dev/tty1** 设备，则 **t_devname** 字段应该包含 **tty1** 。

t_uid 此字段记录了使用此终端设备成功登录的最后一个用户的用户 ID。

t_logtime 此 **time_t** 字段记录了最后一次成功登录到该终端设备的时间。

t_unsuctime 此 **time_t** 字段记录了最后一次不成功登录该终端设备的时间。

t_failures 此字段记录了到该终端设备连续的登录尝试不成功的次数。

t_maxtries 此字段指定了在该终端被锁定前允许的登录尝试连续不成功的次数上限。一旦该终端被锁定，它就必须由被授权的管理员解锁。

t_login_timeout 此字段指定了登录超时值（秒）。

t_logdelay 此字段指定登录重试之间的延迟（秒）。

t_lock 此标志字段表示该终端设备是否在管理上被锁定。此字段只由授权的管理员处理。

举例

下面是终端控制数据库条目的举例：

```
console:t_devname=console:\
    :t_uid=reese:t_logtime#675430072:\
    :t_unsuctime#673610809:\
    :t_maxtries#777:\
    :chkent:
```

此条目针对系统控制台设备， **/dev/console** 。最近的成功登录会话来自用户 **reese** 。该条目记录了当前成功登录的系统时间和最近登录尝试不成功的时间。

警告

远程终端 (ptys) 不应该添加到 **devassign** 或 **ttys** 数据库中。由登录视为 ptys 的设备名格式如下：

ptym/*

pts/*

pty/*

pty[*x*][*y*] 其中， *x* 是一个字母，而 *y* 是一个十六进制数

tty[*x*][*y*] 其中， *x* 是一个字母，而 *y* 是一个十六进制数

telnet/*

作者

ttys 由 HP 开发。

文件

/tcb/files/ttys 终端控制数据库文件

另请参阅

login(1)、 getprtcent(3)、 devassign(4)、 authcap(4)、 default(4)。

名称

ttytype - 终端端口类型数据库

概要

/etc/ttytype

说明

ttytype 是一个标识终端类型的数据库，该终端附加于系统的每个 tty 端口。该文件针对每个端口包含一行、每行包含终端类型（作为 *terminfo*(4) 中列出的名称）、一个空格和一个 tty 设备文件名（不包括开头的 **/dev/** ）。例如，对于 tty02 上的 HP 2622 终端：

2622 tty02

该信息被 **tset** 和 **login** 读取（对于远程登录），用以在登录时初始化 **TERM** 变量（请参阅 *tset*(1) 和 *login*(1) ）。

作者

ttytype 由加州大学伯克利分校开发。

另请参阅

login(1)、 tset(1)。

警告

一些行仅被指定为 **dialup** 或 **plugboard** 。

名称

tunefstab - VxFS 文件系统调整参数表

说明

tunefstab 文件包含 VxFS 文件系统的调整参数。 **tunefs** 为进程命令行选项或在 **tunefstab** 文件中读取参数挂接的文件系统设置调整参数。

tunefstab 中的每个条目是按照以下格式中的一种的字段行：

block-device *tunefs-options*
system-default *tunefs-options*

block-device 是挂接文件系统的设备名称。如果有多个行均指定设备的选项，则会按顺序处理每行并设置选项。

在 *block-device* 的位置， *system-default* 为每个要处理的设备指定了可调整的选项。如果同时有 *block-device* 和 *system-default* 的条目，则 *system-default* 值优先。

tunefstab 中以井字符 (#) 开头的行作为注释并被忽略。

tunefs-options 相应的可调整参数 **vxtunefs** 和 **mount_vxfs** 在文件系统上设置。在此列表中的每个选项是一个 *name*=*value* 对。通过逗号分隔这些选项，选项和逗号之间没有空格或制表符。

有关支持的选项的描述，请参阅 *vxtunefs*(1M) 联机帮助页。

举例

如果您有四列条带化的卷， **/dev/vx/dsk/datadb/db03**，每个磁盘有一个 128 千字节的条形单元，分别将 **read_pref_io** 和 **read_nstream** 参数设置为 128 和四。可以采用以下两种方法之一来执行该操作：

/dev/vx/dsk/datadg/db03 read_pref_io=128k,read_nstream=4

或：

/dev/vx/dsk/datadg/db03 read_pref_io=128k
/dev/vx/dsk/datadg/db03 read_nstream=4

发现将直接 I/O 大小设置为更低从而使它总是低于缺省值，则将下面的行添加到 **/etc/vx/tunefstab** 文件：

/dev/dsk/c3t1d0 discovered_direct_iosz=128K

文件

/etc/vx/tunefstab 缺省的 **tunefstab** 文件。

另请参阅

mkfs_vxfs(1M)、 mount(1M)、 vxtunefs(1M)。

《VERITAS File System Administrator’s Guide》。

名称

tztab - date 和 ctime() 的时区调整表

说明

/usr/lib/tztab 文件描述国际标准时间 (UTC) 和本地时间之间的差异。几个本地区域可以用历史细节同时表示。

文件 **tztab** 由一个或更多时区调整条目组成。条目的第一行包含用户环境中匹配 **TZ** 字符串的值的一个唯一字符串。格式为 **tzname***diff* **dstzname** ，其中 **tzname** 为时区名称或缩写， *diff* 为以小时表示的与 UTC 的差异，另外， **dstzname** 为“Daylight Savings”时区的名称或缩写。 *diff* 的小数值应该在前面加上冒号以分钟表示。每个这样的字符串以字母字符开头。

每个条目的第二及后续行详述该时区的时区调整。这些行各自包含七个字段。其中前六个字段指定时区调整的第一分钟，第七个字段指定其应用。这些字段由空格或制表符分隔。前六个字段为整数模式，指定分钟 (0-59)、小时 (0-23)、日期 (1-31)、月份 (1-12)、年份 (1970-2038)、和星期（0-6，0=Sunday）。一年中的分钟、小时和月份必须包含如上所示（各自）范围中的一个数字。日期、年份和星期可以包含一个数字或由一个减号分隔（表明包括上限和下限的范围）的两个数字。日期和星期字段都必须是一个范围，其他的必须是简单数字。

第七个字段是一个描述时区调整的字符串，用的是它最简单的格式： **tzname***diff* ，其中 **tzname** 是一个给出时区名称或缩写的字母字符串， *diff* 是以小时表示的与 UTC 的差异。 **tzname** 必须与时区调整条目第一行中的 **tzname** 或 **dstzname** 字段匹配。任何小数 *diff* 表示为分钟。

注释以 **#** 开头于 *first* 列中，并且包括换行符之前的所有字符。忽略注释。

如果 **TZ** 字符串的值与表中的任意行都不匹配，那么它按照当前美国模式进行解释。

外部语言环境影响

国际代码集支持

支持单字节字符代码集。

举例

美国东部时区的时区调整表为:

```
EST5EDT
0 3 6     1  1974      0-6 EDT4
0 3 22-28 2  1975      0   EDT4
0 3 24-30 4  1976-1986 0   EDT4
0 3 1-7   4  1987-2038 0   EDT4
0 1 24-30 11 1974      0   EST5
0 1 25-31 10 1975-2038 0   EST5
```

通常（如第一行中所示）东部标准时间比 UTC 早五个小时。 Daylight Savings 时间中，它保持 4 小时差异的变化。 Daylight Savings 时间生效（第二行）的第一时间是在 1974 年 1 月 6 日上午 3 点，EDT。请注意，之前的分钟是上午 1:59，EST。更改回的标准时间生效（第六行）于同年九月份的最后一个星期日。此时，时间经过从上午 1:59，EDT 到上午1:00，EST。其后 Daylight Savings 时间的转换经历了从二月的最后一个星期日（第三行）到四月的最后一个星期日（第四行）再到四月的第一个星期日（第五行）。同时期标准时间的返回保持在十月的

最后一个星期日（第七行）。

作者

tztab 由 HP 开发。

文件

/usr/lib/tztab

另请参阅

date(1)、 ctime(3C)、 environ(5)。

名称

ups_conf - HP PowerTrust 不间断电源系统 (UPS) 监视器配置文件

说明

HP PowerTrust UPS 监视器守护程序的缺省配置文件 (**ups_mond**) 。只要符合指定的格式，其他文件也可以使用，并且监视器守护程序配置为使用备用文件（请参阅 *ups_mond*(1M) 中的 **-f** 选项的描述）。

配置文件中的行最多可以包含 256 个字符，另外 UPS-tty 设备文件的完整路径名最多可以包含 100 个字符。

配置文件中每行只允许有一个条目。每一行以一个关键字开头。配置文件条目中的所有字段用冒号 (:) 分隔。配置文件中的条目以遇到的第一个空格结束（正如库函数 **isspace()** 指定的那样；请参阅 *isspace*(3C) ）。每行中第一个空格之后的字符被视为注释。

关闭延迟和超时的值应该在文件中的第一行。

文件中的条目以如下所示的关键字开头。终止冒号将关键字与它的参数的值分隔开。 **ups_mond** 可识别下列关键字:

shutdown_delay_mins
: 在 **ups_mond** 初始化 **shutdown -h** 之前，其相应的 UPS 正在使用内部电源（电池）运行的第一个 **upstty** 通知的分钟数。缺省值为一分钟。如果它们对站点来说是相同的，这个值应该设置为代表瞬间电源中断。

shutdown_timeout_mins
: 在使用暂停选项（ RB_HALT ；请参阅 *reboot*(2) ）启动 **reboot** 之前，监视 **shutdown -h** 运行的分钟数。缺省值为五分钟。这个值要比执行 *shutdown*(1M) 需要的最长时间更长。请注意，在 **shutdown_timeout_mins** 的时间过去后，AC 线电压不足的 UPS 将关闭。一旦恢复 AC 线电压，UPS 将恢复它的输出功能。这个超时值不应该比可能会让观测者变得不耐烦的 **shutdown** 更长。请注意，重要的是，这个值是 UPS 延迟它的电源重新接通的时间段，即使 AC 电源很快恢复。

kill_after_shutdown
: 这一行后接 **Y** 或 **N** ，它分别指出 AC 线电压失败时 UPS 需要关闭 (**kill_after_shutdown:Y**) 或打开 (**kill_after_shutdown:N**) 。缺省值为 **kill_after_shutdown:N** 。

upstty
: 通过配置 UPS 的 **tty** 设备专有文件的完整路径名。包括每个 UPS 的一个条目。 **upstty** 条目是按照它们出现在 **/etc/ups_conf** 中的顺序处理的。因此，重要的是按优先顺序列出不间断电源系统（例如，保护 SPU 的 UPS 列在首位）。

  upstty 条目可以包含下列可选参数，它可以如下列 **upstty** 设备专用文件名的任意顺序出现:

MSG_ONLY
: 此 **upstty** 将不会启动 **shutdown** 或 **reboot** 。

SOLA
: 不间断电源供应的类型。目前，这仅是定义过的类型和支持 HP UPS 的系列，包括 PowerTrust。它缺省为 SOLA 。

举例

下面是 **/etc/ups_conf** 文件的示例:

```
shutdown_delay_mins:1
shutdown_timeout_mins:5
kill_after_shutdown:Y
upstty:/dev/tty0p1
upstty:/dev/tty0p2:MSG_ONLY
upstty:/dev/tty0p3:SOLA:MSG_ONLY
upstty:/dev/tty0p5:SOLA
```

文件

/dev/tty*

/etc/ups_conf

另请参阅

ups_mond(1M)。

名称

utmp、 wtmp、btmp - utmp、wtmp、btmp 条目格式

概要

#include <sys/types.h>

#include <utmp.h>

说明

这些文件存放 **last** 、 **who** 、 **write** 和 **login** 命令的用户和记账信息（请参阅 *last*(1) 、 *who*(1) 、 *write*(1) 和 *login*(1) ），其结构由 **<utmp.h>** 定义，如下：

```
#define UTMP_FILE               "/etc/utmp"
#define WTMP_FILE               "/var/adm/wtmp"
#define BTMP_FILE               "/var/adm/btmp"
#define ut_name        ut_user

struct  utmp {
  char   ut_user[8];            /* User login name */
  char   ut_id[4];              /* /etc/inittab id(usually line#)*/
  char   ut_line[12]            /* device name (console, lnxx) */
  pid_t  ut_pid;                /* process id */
  short  ut_type;               /* type of entry */
  struct exit_status
     short  e_termination;      /* Process termination status*/
     short  e_exit;             /* Process exit status*/
     } ut_exit;                 /* The exit status of a process*/
                                /* marked as DEAD_PROCESS.*/
  unsigned short ut_reserved1;  /* Reserved for future use*/
  time_t  ut_time;              /* time entry was made*/
  char    ut_host[16];          /* host name,if remote*/
  unsigned long ut_addr;        /* host Internet addr, if remote*/
};

/*  Definitions for ut_type  */
#define EMPTY                   0
#define RUN_LVL                 1
#define BOOT_TIME               2
#define OLD_TIME                3
#define NEW_TIME                4
#define INIT_PROCESS            5    /* Process spawned by "init" */
#define LOGIN_PROCESS           6    /* getty process awaiting login */
```

```
#define USER_PROCESS      7      /* A user process */
#define DEAD_PROCESS      8
#define ACCOUNTING        9
#define UTMAXTYPE         ACCOUNTING      /* Max. legal value of ut_type */

/*  Special strings or formats used in the "ut_line" field */
/*  when  accounting for something other than a process  */
/*  No string for the ut_line field can be more than */
/*  11 chars + a NULL in length  */
#define RUNLVL_MSG        "run-level %c"
#define BOOT_MSG          "system boot"
#define OTIME_MSG         "old time"
#define NTIME_MSG         "new time"
```

utmp 文件包含登录到系统中的所有用户记录。 **btmp** 文件包含每一个无效登录尝试的错误登录条目。 **wtmp** 文件包含所有登录和注销记录。

注意， **wtmp** 和 **btmp** 趋向于无限增大，应定期对其进行检查。不再有用的信息应定期删除，从而防止其变得过大。还要注意， **wtmp** 和 **btmp** 不能由维护它们的程序创建。因此，如果这些文件已删除，登录记录保留被关闭。

文件

/etc/utmp
/var/adm/wtmp
/var/adm/btmp

作者

utmp 、 **wtmp** 和 **btmp** 由 HP 和加州大学伯克利分校联合开发。

另请参阅

last(1)、 login(1)、 who(1)、 write(1)、 acctcon(1M)、 fwtmp(1M)、 utmpd(1M)、 getut(3C)、 getuts(3C)、 getutx(3C)。

符合的标准

<utmp.h>: XPG2

名称

utmps - 用户记账数据库

概要

#include <sys/types.h>

#include <utmps.h>

说明

utmps 文件包含所有登录到系统的用户记账信息。仅当 *utmpd*(1M) 未运行时， *getuts*(3C) 才能访问该文件。下面信息存储于 **utmps** 文件中：

- 用户登录名（最多 256 字符）
- **/etc/lines** ID
- 设备名（控制台， *lnxx* ；最多 64 个字符）
- 进程 id
- 条目类型
- 被标记为 **DEAD_PROCESS** 的进程退出状态
- 条目生成时间
- 主机 Internet 地址，如为远程（支持 IPv4 和 IPv6 地址）。
- 主机名，如为远程（最多 256 个字符）

仅当 *utmpd*(1M) 未运行时，该文件包含用户记账信息。当 *utmpd*(1M) 激活时，该文件不包含最新的用户记账信息。在未来版本可能不再使用该文件。

文件

/etc/utmps

作者

utmps 由 HP 开发。

另请参阅

last(1)、 login(1)、 who(1)、 write(1)、 acctcon(1M)、 fwtmp(1M)、 utmpd(1M)、 getuts(3C)。

名称

utmpx - utmpx 数据库存储文件

概要

#include <sys/types.h>

#include <utmpx.h>

说明

utmpx 文件包含所有登录到系统的用户记账信息。使用此文件，而不是正被淘汰的 **utmp** 文件。下面信息存储于 **utmpx** 文件中：

- 用户登录名（最多 24 个字符）
- /etc/lines id
- 设备名（控制台，lnxx）
- 进程 id
- 条目类型
- 被标记为 DEAD_PROCESS 的进程的退出状态
- 条目生成时间
- 主机的 Internet 地址，如为远程

当前 HP-UX 版本升级 **utmp** 和 **utmpx** 文件及格式。不建议直接使用 **utmpx** 文件， **utmp** 和 **utmpx** 文件必须同步更新。该功能由 libc API **pututline** 和 **pututxline** 提供

文件

/etc/utmpx

作者

utmpx 由 HP 和加州大学伯克利分校联合开发。

另请参阅

last(1)、 login(1)、 who(1)、 write(1)、 acctcon(1M)、fwtmp(1M)、 utmpd(1M)、 getut(3C)、 getuts(3C)、 utmp(4)。

符合的标准

<utmp.h>： X/OPEN 4.2

名称

uuencode - 编码 uuencode 文件的格式

说明

uuencode 输出的文件由一个后面有几行正文的标题行，以及一个结尾行组成。 **uudecode** 命令忽略标题之前或结尾之后的所有行（请参阅 *uuencode*(1)）。标题之前的行必须与标题不相象。

标题行的组成包括单词 **begin** 、之后的空格、模式（八进制）、再一个空格以及指定远程文件名的字符串。

正文由一些行构成，每行包含 62 个或更少的字符（包括结尾的换行符）。这些行由字符计数、编码字符和换行符组成。

字符计数是一个打印字符，该字符代表一个整数。这个整数是该行剩余部分的字节数，范围总是从 0 到 63。该字节数可以通过从字符中减去 ASCII 空格字符的等效八进制值（八进制数 40）来决定。

3 个字节的组保存在 4 个字符中，每个字符 6 个比特。所有都通过空格来补位，使字符可以打印。最后一行可能小于正常的 45 个字节。如果大小不是 3 的倍数，则该值可以通过最后一行的计数值决定。如果需要，可以包含额外无意义的数据，来使字符数是 4 的倍数。正文以计数为零的行终止。该行由一个 ASCII 空格组成。

末尾行由一个单独的单词 **end** 组成。

另请参阅

mail(1)、 uuencode(1)、 uucp(1)。

名称

wtmps、btmps - 用户登录信息

概要

#include <sys/types.h>

#include <utmps.h>

说明

wtmps 和 **btmps** 保存为如 **last** 、 **who** 、 **write** 和 **login** 命令的用户和记账信息（请参阅 *last*(1) 、 *who*(1) 、 *write*(1) 和 *login*(1) ）。

btmps 文件包含每个无效登录尝试的错误登录条目。 **wtmps** 文件包含所有登录和注销的一个记录，记账记录除外。这些文件包含类似 **utmps** 结构，这个结构的关键元素如下给出：

char ut_user[]	用户登录名
char ut_id[]	区分条目的唯一 ***ID***
char ut_line[]	设备名
pid_t ut_pid	进程 ***Id***
short ut_type	条目类型
struct ut_exit	进程的退出状态
struct timeval ut_tv	生成时间条目
char ut_host[]	主机名，如为远程
uint8_t ut_addr[]	主机的 ***Internet*** 地址， 如为远程
short ut_addr_type	标识地址类型的标志 ***in ut_addr***

```
#define WTMPS_FILE        "/var/adm/wtmps"
#define BTMPS_FILE        "/var/adm/btmps"
```

注意， **wtmps** 和 **btmps** 趋向于无限增大，应定期检查。不再有用的信息应定期删除，从而防止文件变得过大。还应注意 **wtmps** 和 **btmps** 不能由维护它们的程序创建。因此，如果这些文件被删除，登录记录保留被关闭。

作者

wtmps 和 **btmps** 由 HP 开发。

文件

/var/adm/wtmps

/var/adm/btmps

另请参阅

last(1)、 login(1)、 who(1)、 write(1)、 acctcon(1M)、 fwtmp(1M)、 wtmpfix(1M)、 getuts(3C)。

名称

ypfiles - 网络信息服务数据库和目录结构

说明

备注

网络信息服务 (NIS) 以前称为黄页 (YP)。尽管名称已改变，但该服务的功能仍与过去相同。

网络信息服务 (NIS) 的网络查找服务使用 **/var/yp** 下以目录分层结构显示的数据库。这些数据库仅存在于用作 NIS 服务器的计算机上。一个数据库包含两个由 **makedbm** （请参阅 *makedbm*(1M) ）创建的文件。一个文件的扩展名是 **.pag** ，另一个文件的扩展名是 **.dir** 。例如，名为 **netgroup** 的数据库由 **netgroup.pag** 和 **netgroup.dir** 文件对实现。 NIS 使用的数据库称为 NIS *map* 。

一个 NIS *domain* 就是一个指定的网络信息服务映射集。每个 NIS 域都是作为 **/var/yp** （其名称就是域名）的一个子目录进行工作的，而且每个 NIS 域中都包含该域的映射。 NIS 域的数目可以是任意多个，且每个域中都可包含任意多个映射。

除了在 **/var/yp/***domain* 中包含的这些数据库之外， NIS 主服务器还包含名为 *general_NIS_mapname***.time** 的多个文件。这些文件仅仅是一些空文件，它们的上次修改时间将与从中创建映射的 ASCII 文件的上次修改时间进行比较。 **ypmake** 脚本执行这些比较操作，从而确定这些映射是否是当前映射（请参阅 *ypmake*(1M) ）。在下面的 文件部分中进一步描述了 *general_NIS_mapname* 名称。

尽管在系统其他部分的操作中可能会需要映射，但 NIS 查找服务并不需要映射。 NIS 服务器对其提供访问的映射列表，既不受限制，也不必将所有映射都包含其中。如果一个映射位于一个给定的域中，且有一个客户端询问此映射的信息，则 NIS 会处理此种情况。一个映射要想是可始终访问的，则它必须位于所有服务该域的 NIS 服务器中。要使复制映射间的数据保持一致，请在每一个服务器根目录下的 **crontab** 文件中创建一个条目，用以定期运行 **ypxfr** （请参阅 *ypxfr*(1M) 和 *crontab*(1) ）。在 *yppush*(1M) 和 *ypxfr*(1M) 中有关于此主题的更多信息。

NIS 映射中包含两个特殊的键值对。第一个键是 *NIS_LAST_MODIFIED* ，它的值是一个 10 字符的 (ASCII) 顺序编号。顺序编号是创建该映射时的 **time()** ，以秒为单位（请参阅 *time*(2) ）。第二个键是 *NIS_MASTER_NAME* ，它的值是该映射的主 NIS 服务器的主机名。 **makedbm** 命令自动生成这两个键值对。当 **ypxfr** 命令从一个 NIS 服务器向另一个服务器传输映射时，将使用这些值。

仅能在主服务器上生成和修改 NIS 映射。使用 **ypxfr** 将这些映射复制到从服务器中，从而使运行在不同体系结构计算机上的各 NIS 服务器避免出现潜在的字节序问题，并最大限度地减小数据库所需的磁盘空间（参阅 *ypxfr*(1M)）。使用 **ypinit** ，最初可为主服务器和从服务器创建 NIS 数据库（请参阅 *ypinit*(1M) ）。

当服务器的数据库创建以后，一些映射的内容就会发生改变。通常，每个数据库的 ASCII 源版本位于主服务器中，并可通过文本编辑器来进行更改。 NIS 映射会重建以包括这些变动，并可通过运行 **ypmake** shell 脚本由主服务器传送到从服务器中（请参阅 *ypmake*(1M) ）。

在 **ypmake** 脚本或 NIS **Makefile** 中包含的命令可创建所有的 NIS 映射。若添加一个非标准的 NIS 映射，则编辑 **ypmake** 脚本或 **Makefile** 以支持该新映射（标准的 NIS 映射将在 文件下面讨论）。 **ypmake** 和 **Makefile** 使用 **makedbm** 以在主服务器上生成 NIS 映射，并可以运行 **yppush** 来将重建的映射复制到从服务器中（请参阅 *yppush*(1M) ）。 **yppush** 命令读取名为 **ypservers** 的映射，该映射包含特定域的所有 NIS 服务器的主机名。有关

详细信息，请参阅 *ypmake*(1M) 、 *yppush*(1M) 和 *ypxfr*(1M) 。

相关内容

如果 **/var/yp** 所在的文件系统不允许文件名超过 14 个字符，而要为网络信息服务创建一个新的非标准映射，则该映射的名称长度必须不超过 10 个字符。此规则存在是因为 **makedbm** 会为任何一个映射名添加 4 字符的后缀 **.dir** 和 **.pag** 。

下表描述了标准的 NIS 映射名到存储在 14 字符文件名文件系统上的较短名称的转换。不管哪一台计算机是 NIS 服务器，标准的映射名应由 HP 计算机上的 NIS 客户端在进行请求时使用。

标准的 **NIS** 映射名	映射名缩写
mail.aliases	***mail.alias***
mail.byaddr	***mail.byad***
ethers.byaddr	***ether.byad***
ethers.byname	***ether.byna***
group.bygid	***group.bygi***
group.byname	***group.byna***
hosts.byaddr	***hosts.byad***
hosts.byname	***hosts.byna***
netgroup	***netgroup***
netgroup.byhost	***netgr.byho***
netgroup.byuser	***netgr.byus***
netid.byname	***netid.byn***
networks.byaddr	***netwk.byad***
networks.byname	***netwk.byna***
passwd.byname	***passw.byna***
passwd.byuid	***passw.byui***
protocols.byname	***proto.byna***
protocols.bynumber	***proto.bynu***
publickey.byname	***pbkey.byna***
rpc.byname	***rpc.byna***
rpc.bynumber	***rpc.bynu***
services.byname	***servi.byna***
auto.master	***auto.mast***
ypservers	***ypservers***

作者

ypfiles 由 Sun Microsystems, Inc. 开发。

文件

下表提供了标准的网络信息服务映射信息。

General NIS Mapname 列列出 NIS 映射集的名称；这些集合中包括 *Standard NIS Mapname* 列中的相邻条目。

ASCII Source 列列出了在 HP 主 NIS 服务器上从中创建映射的 ASCII 文件。 **ypmake** 脚本允许源目录或口令映射的文件发生变化。

Standard NIS Mapname 列列出各名称，通过这些名称，映射可存储在 NIS 服务器上，并可被 NIS 客户端引用。

常规 NIS Mapname	ASCII 源	标准 NIS Mapname
aliases	***/etc/mail/aliase***	***mail.aliases***
		mail.byaddr
ethers	*	***ethers.byaddr***
		ethers.byname
group	***/etc/group***	***group.byname***
		group.bygid
hosts	***/etc/hosts***	***hosts.byname***
		hosts.byaddr
netgroup	***/etc/netgroup***	***netgroup***
		netgroup.byhost
		netgroup.byuser
netid	***/etc/netid***	***netid.byname***
networks	***/etc/networks***	***network.byaddr***
		network.byname
passwd	***/etc/passwd***	***passwd.byname***
		passwd.byuid
protocols	***/etc/protocols***	***protocols.byname***
		protocols.bynumber
publickey	***/etc/publickey***	***publickey.byname***
rpc	***/etc/rpc***	***rpc.byname***
		rcp.bynumber
services	***/etc/services***	***servi.bynp***
		services.byname
automounter	***/etc/auto_master***	***auto.master***
ypservers	***	***ypservers***

* 这些数据库不在 HP 网络信息服务主服务器上创建。然而，如果 HP 计算机是用来创建和分布这些数据库的主 NIS 服务器的一个从服务器，则此 HP 从 NIS 服务器将存储这些数据库。如果您有一个非 HP 计算机需要这些映射，则建议将该计算机设置为主 NIS 服务器。设置过程中，这些映

射应根据需要进行创建。

*** **ypservers** 数据库没有 ASCII 源。该源是从由主 NIS 服务器上 **ypinit** 的用户所提供的响应中创建的，且它没有匹配的 **ypservers.time** 文件。

另请参阅

domainname(1)、 makedbm(1M)、 rpcinfo(1M)、 ypinit(1M)、 ypmake(1M)、 yppoll(1M)、 yppush(1M)、 ypserv(1M)、 ypxfr(1M)。